TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2020

42

Mª del Carmen Rodríguez Hernández

Context-Aware Recommendation Systems in Mobile Environments

Departamento
Informática e Ingeniería de Sistemas

Director/es
Ilarri Artigas, Sergio

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

CONTEXT-AWARE RECOMMENDATION SYSTEMS IN MOBILE ENVIRONMENTS

Autor

Mª del Carmen Rodríguez Hernández

Director/es

Ilarri Artigas, Sergio

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2017

Context-Aware Recommendation Systems in Mobile Environments

María del Carmen Rodríguez Hernández

Tesis Doctoral

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Universidad Zaragoza

October 2017

I dedicate this thesis to all
those who supported and guided me
to make one of my dreams come true.

Acknowledgements

This thesis is the result of several years of hard work, during which I gave up many things to grow professionally and received the support of many people. The following lines are a sign of my infinite gratitude to all of them.

Firstly, I would like to thank my supervisor, Sergio Ilarri, who was also my mentor during all these years. I sincerely thank his support, dedication, emails, deep revisions, useful comments, and invaluable ideas, that helped me to improve this thesis and develop the published works. From him I have learned to do research, to analyze all the details, and find logical explanations behind the experiments. This thesis would not have been possible without his help and constant supervision. It has really been a pleasure to work with him.

I would like to express my profound gratitude to Ramón Hermoso and Raquel Trillo, for their encouragement and collaboration in different parts of this thesis. A special thanks also to Francesco Guerra for the work we did at the University of Modena and Reggio Emilia (Italy), which was essential to contribute in the field of context-based recommendations and keyword-based search in mobile environments. I must say thanks to Sergio Cleger Tamayo, Juan Francisco Huete Guadix, and Juan Manuel Fernández Luna. They were who introduced me to the field of Recommendation Systems, and they contributed to keep my motivation through the research path. Additionally, I would like to thank the anonymous reviewers of the papers that have been submitted in various journals and conferences, who have provided valuable feedback that improved the quality of this work. I would also like to acknowledge the support of the CICYT projects TIN2016-78011-C4-3-R (AEI/FEDER, UE), TIN2013-46238-C4-4-R, TIN2010-21387-C02-02, and DGA-FSE, as well as the Keystone (semantic KEYword-based Search on sTructured data sOurcEs) COST Action IC1302, and a Banco Santander scholarship.

I also owe thanks to people with whom I shared meetings, discussions, and nice moments at the University of Zaragoza: José Ignacio, Roberto, Óscar, Diego, Emanuele, Ricardo, José, Simona, Carlos, and Andrés. I want to thank all the Cuban people I met at the University of Zaragoza (Irina, Idelkys, Yaneisy, Dayany, Yanet, Venelio, Anamaris, and Alain). I am also very grateful to the staff of the International Relations Section of the University of Zaragoza, who made it easier for me to meet the rest of the members of the Banco Santander scholarship. Special thanks to Eva, Asun, and Araceli, who always had time to encourage me and share pleasant moments of

coffee, Cuban dinner, and dance.

My sincere gratitude to distinguished professors of the University of Holguín (Cuba), who educated me and created the solid basis for carrying out this research. Special thanks to Rosa, Sergio, Jenny, Ángel, Luis C., Matilde, Ana de Lourdes, Leydis, Rita, Félix, Rodolfo, Yoisel C., Pedro, and Irma. I also owe thanks to the rest of the Department of Computer Science, for all the times we shared at the University of Cuba: Ángel, Julio C., Scull, Yuliet, Sandy, Mario, Marlen, Liset, Miday, Yisel, Eduardo, Leandro, Luis, Ricardo, Reynol, Dalilis, Yisel C., Roberto B., Rafael, Frank, Daymy, and Humberto. They are all my friends too.

I want to thank all of my friends, who showed me that there is life outside my academic bubble. My lifelong friends, Yuri, Amado, Marlen, Maité, Miday, Liset, Yisel, Valentín, Alejandro, Ángel, Nadiezda, Julio C., Henry, Haydé, Yanet, Magela, Yadira, Tania, Lianet, Linnet, Analilian, Kenia, Marco, and Armelio, with whom I shared unforgettable moments (parties, dances, discussions, excursions, carnivals, etc.). My dear galician friends, Montse and Olga, who let me enter with much affection in their lives and with which I enjoy their sincere discussions and company. Thanks to them, I also inherited their valuable friends from Zaragoza: Göknur, Enrique, Marco, Logan, Paul, Chuan, Willy, Louis, Luilli, Gregorio, and María. I also want to thank the support, encouragement, understanding and wise advice of Noemí, Armando, David, and Carol.

Above all, I would like to express my profound gratitude to my dear family, especially my parents José Antonio and Margarita, and my sister María Margarita, for their infinite love, guide, understanding, support, encouragement, and patience. My achievements are also yours. Special thanks goes to my grenadian David, for his love, unconditional support, company and patience during all these years. Thank you for all the things I have learned with you. My gratitude also goes to my second family: Paco, Carmen, Lola, Jose, Guillermo, Lucía, María del Pilar, Carlos, and Javier. Thanks for their talks, encouragement, concern, and wise advices.

In general, I thank all those who trusted me and taught me to grow in the professional, scientific, and personal fields. I dedicate my work to all of them.

Zaragoza, October 2017
María del Carmen Rodríguez Hernández

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, the huge amount of information available may easily overwhelm users when they need to take a decision that involves choosing among several options. On the one hand, it is necessary to identify which items are relevant for the user at a particular moment and place. On the other hand, some mechanism would be needed to rank the different alternatives. *Recommendation Systems* (*RS*) [RV97, RRSK11, LMY+12, BOHG13, LRU14], that offer relevant items to the users, have been proposed as a solution to these problems. The main goal of these systems is to recommend certain items based on user preferences. The use of RS has increased in different application scenarios [LWM+15]. For example, they have been proposed for the recommendation of books, music, movies, news, friends in social networks, and even traffic signal timings [ZGWW17]. Netflix [HR97, Ama13, GUH16, Aro16], Amazon [Bez94, LSY03, Aro16, SL17], MovieLens [Gro96], TripAdvisor [KS+00, WCN12] and IMDb [Jay90] are examples of popular recommendation applications that currently play an important role in the Internet.

Most RS operate in a two-dimensional (2D) $User \times Item$ space. However, considering only information about the users and items is not enough in applications such as the recommendation of vacation packages, where the recommendation system should be able to suggest places appropriate, for example, for the summer or winter season. In this case, it is important not only to determine which items should be recommended, but also when these recommendations should be provided and how to combine them in a ranked list. Therefore, additional contextual information (e.g., season, time, weather, location, etc.) should be considered in the recommendation process.

Examples like the one above have motivated research on *Context-Aware Recommendation Systems* (*CARS*) [ASST05, AT11], which additionally consider contextual information related to the collected preferences ($User \times Item \times Context$ space). Thus, CARS can discriminate the interest of users about particular items in several situations.

Moreover, the context of a user in a mobile computing scenario is highly dynamic

(e.g., the location of the user usually changes constantly). Therefore, recommendation algorithms should be able to effectively and efficiently exploit the dynamic context of the user in order to offer her/him suitable recommendations and keep them up-to-date.

The research area of this thesis belongs to the fields of context-aware recommendation systems and mobile computing. We focus on the following scientific problem: *how could we facilitate the development of context-aware recommendation systems in mobile environments to provide users with relevant recommendations?* This work is motivated by the lack of generic and flexible context-aware recommendation frameworks that consider aspects related to mobile users and mobile computing. We attempt to make a step forward in that direction and encourage further research in this area.

In order to solve the identified problem, we pursue the following general goal: *the design and implementation of a context-aware recommendation framework for mobile computing environments that facilitates the development of context-aware recommendation applications for mobile users.* In this thesis, we contribute to bridge the gap not only between recommendation systems and context-aware computing, but also between CARS and mobile computing.

In order to guide the research, we focus on four main aspects, which are summarized below:

1. *What is the technological background related to context-aware recommendations in mobile environments?*

2. *How could we develop data management techniques that enable the deployment of a context-aware recommendation system for mobile environments?*

3. *How could we determine the potential interest of these types of context-aware mobile recommendations?*

4. *How could we evaluate the architecture proposed for the validation of the results in several application domains?*

Throughout the thesis, we will try to answer each of the previous scientific questions. For that purpose, we propose a specific framework called *MOONRISE* (*MObile cONtext-aware RecommendatIon SystEm*). It is a generic and flexible context-aware recommendation architecture that aims to facilitate the development of context-aware recommendation systems in mobile computing environments. In order to improve the accuracy of the recommendations and to solve the *cold start* and *sparse data* problem, the architecture allows the hybridization of several strategies for recommendation (e.g., content-based filtering and collaborative filtering), as well as the possibility to exploit data available outside the local knowledge base obtained through a mobile peer-to-peer (P2P) network to obtain missing useful information. In addition, the architecture accommodates different context-aware recommendation paradigms (*pre-filtering*, *post-filtering*, and *contextual modeling*). Both *pull-based recommendations* (reactive recommendations, obtained as an answer to a query submitted by the user

and evaluated by the system as a continuous query) and *push-based recommendations* (proactive recommendations, received without explicit requests from the user) are supported. In both cases, appropriate recommendations are provided to the user by taking into account (static and dynamic) contextual information in real-time.

In this chapter, we firstly summarize the technological context necessary to facilitate the understanding of the problem of context-aware recommendations in mobile environments. Secondly, we present the motivation of our thesis. Thirdly, we provide a general overview of the main contributions presented in this thesis. Finally, we describe the structure of the thesis.

1.1 Context of the Thesis

The research context of this thesis is centered on context-aware recommendation systems and mobile computing. Specifically, we focus on the existing gap between context-aware recommendation systems and mobile computing.

1.1.1 Mobile Computing

Mobile computing provides flexible communication between people, as well as continuous access to data and network services *anywhere and at anytime*. In this context, the users with portable devices have access to a shared infrastructure independent of their physical location. The main characteristics and challenges of mobile computing can be summarized as follows: [FZ94, Sat96, Sat10]:

- *Wireless communication.* Mobile devices require wireless network access, which are subject to some limitations, such as more frequent disconnections, lower bandwidth, greater variation in the available bandwidth, greater network heterogeneity, and increased security problems.

- *Mobility.* Mobile devices are able to change their location while they are connected to the network. This generates several problems. For example, the network addresses change dynamically and the current location may affect the relevance of data as well as the appropriate answers to user queries and data needs.

- *Portability.* Mobile devices are portable. The convenience of using portable devices introduces limitations, such as low power, more risk of data loss, small user interfaces, and small storage capacity.

In recent years, there has been a great interest in mobile computing research, inspired by the increased use of mobile devices [Sat01, EW17]. The rapid progress of mobile computing is continuous regarding the development of technologies in industry and commerce. Some key hot trends are the development of mobile P2P architectures [IDTL15], Location-Based Services (LBS) [IIMS11, YMII14], and Intelligent Transportation Systems (ITS) [IWD14], among others.

In order to contribute to the improvement of potential performance problems (e.g., reduced battery life, storage capacity, and bandwidth) in mobile computing, *Mobile Cloud Computing (MCC)* [DLNW13, MAV17] has emerged as a potential technology for the development of mobile services. It integrates the concept of cloud computing into the mobile environment and refers to an infrastructure where both the data storage and data processing happen outside the mobile device [YD16]. MCC provides on-demand resources, simplifies the use of hardware, and eliminates some technical barriers related to the mobile devices' performance. However, moving data from and into the cloud is not easy, especially when high bandwidth is required and applications need fast real-time responses.

Rapid advances of communication technologies and mobile devices have led to the emergence of new computing paradigms that alleviate the disadvantages of cloud computing, such as unpredictable latency, bandwidth bottlenecks, lack of mobility support, and location awareness, among other [HNYL17]. This new trend is an extension of cloud computing and it is called *fog computing*, also known as *Mobile Edge Computing (MEC)*, which brings running applications and related processing tasks closer to the mobile users, in such a way that the network congestion is reduced and the applications perform better [AA16, MYZ+17]. Fog computing positively influences domains such as mobile computing, Internet of Things (IoT) [AIM10, GBMP13, WF15], and big data analytics, that could help in reducing latency, increasing throughput, consolidating resources, saving energy, and enhancing security and privacy [HNYL17, YvLJ+17, IPCF17, AAHC17, WYG+17].

The integration of mobile computing with CARS is of great importance for this thesis. It would facilitate the exploitation of the dynamic contextual information of mobile users (or items) in real-time to provide recommendations of high quality in specific situations.

1.1.2 Context-Aware Recommendation Systems (CARS) in Mobile Environments

Recommendation Systems can be described as an information filtering technology that suggests information (or items) to the user, taking into account her/his tastes. Generally, these systems are classified into collaborative filtering and content-based filtering [AT05]. In order to identify the useful items for the user, a recommendation system must be able to estimate the utility (or rating) of each of them, and then decide which items to recommend based on this estimation. For example, a recommendation system for restaurants can be implemented by a classifier [KZP07], which estimates one of five classes (ratings in the scale of one to five) for each item not seen by the current user, based on a number of features that describe it [RRSK11]. The resulting items to recommend can be the K items with the best prediction or those with a predicted rating above a predefined threshold.

Traditional recommendation systems deal with applications having only two dimensions, users and items ($User \times Item$), and do not consider contextual information (e.g., time of the day, company of other people, day of the week, etc.) dur-

ing the recommendation process. However, recent approaches have highlighted the importance of considering the context of the situation in which the recommendation process takes place, in order to offer more relevant and precise recommendations [AT11]. As a consequence, the integration of recommendation systems and context-aware computing have given rise to the so-called context-aware recommendation systems [ASST05, AT08, AMRT11].

Context-aware recommendation systems are traditionally classified in the following paradigms [AT11]: *pre-filtering*, where the contextual information is used to filter the data set before applying traditional recommendation algorithms; *post-filtering*, where the ratings are predicted using a conventional 2D recommendation system, taking all the input data available into account, and then the resulting set of recommendations is adjusted (contextualized) for each user by using contextual information; and *contextual modeling*, which uses the context information directly in the modeling technique as part of the estimation of ratings.

Most existing research in the field of CARS considers only static context information [AT11, KLK16], despite the fact that exploiting dynamic context information would be very helpful in mobile computing scenarios. This implies that the recommendations produced by the system are static. Even if the user's dynamic environment (e.g., the user's location) changes, the data provided to the user are not updated based on the changes generated in the user's dynamic context. Instead, during the recommendation process, only the static context (i.e., context attributes that do not change frequently, such as the information previously entered by the user into the system about her/his age, job, etc.) is taken into account.

Research on context-aware recommendation systems and research on mobile computing often progress independently, with little synergies between them. Hence, a current challenge is the integration of CARS with mobile computing. The emergence of new context-aware recommendation approaches that consider the peculiarities of mobile environments is key to obtain recommendations of high quality. Indeed, the main goal of context-aware mobile recommendations is to suggest the right items (or services) to mobile users anywhere and at anytime, being the contextual information a key element to determine their relevance.

1.2 Motivation

Nowadays, the interest in context-aware recommendation systems has grown due to the need to recommend items where the contextual information is relevant. Moreover, the emergence of mobile computing has opened new possibilities in the field of CARS. For example, traditional RS are typically accessed from a website or desktop application at home, but in recent years, mobile computing has facilitated the development of CARS for mobile devices. Sometimes people want to receive recommendations at any time, no matter where they are. In addition, the context often is location-dependent and sensors (e.g., the Global Positioning System –GPS– receiver and accelerometer) embedded in mobile devices can be exploited to access this type of information, while on a computer at home the location will not change.

However, CARS is still an emerging and underexplored field [RRSK11]. Indeed, Adomavicius and Jannach claimed recently that there is still much research needed in the field of CARS [AJ14]. According to their study, the main research issues studied so far are related to the problem of understanding and representing the context in recommendation systems, the development of different recommendation algorithms that include contextual information, and the evaluation of context-aware recommendation paradigms [PTG14b]. Despite these efforts, the design of flexible and generic architectures and frameworks to support an easy development of CARS has been relatively unexplored. Particularly, in mobile environments, where the user is moving and the context is highly dynamic, aspects related to mobile users and mobile computing are usually ignored in the field of CARS. Among others, we would like to highlight the following problems identified, that motivated our research:

- *Centralized architecture.* One common feature of CARS is that they follow a centralized (client-server) topology [BBC+08, LCVA12, BHD13, CMVGRG+15, PGPS16, LT16]. The main limitations of this architecture are related to the bottleneck generated in the server, which must be always running and accessible by the clients during the recommendation process. Recently, a few attempts to develop decentralized CARS have been performed [YYN10, YYN12]. This has been demonstrated by an increasing attention to P2P systems, which are characterized by a cost reduction, improved scalability, reliability and robustness, dynamism, and high levels of anonymity and privacy [Ric10].

- *Specific scenarios.* Most proposed approaches are designed and built for specific scenarios of CARS, such as tourism [SPK04, GLX+11, MLCM13, GKMP14, BRLW15, CPGPSM16], news [SJM14, PCV+16], movies [ONM+12, CFTCD13, CMVGRG+15], music [BKL+11, WRW12, BDH13, BHD13], etc. They are ad hoc solutions, which limits the extensibility of CARS to other application domains.

- *Static context.* Most CARS consider a static representation of context [AT11, KLK16]. However, in mobile environments, where the context can change very quickly due to the mobility of the users and/or items involved, context-aware recommendation systems should be able to update continuously (e.g., at a certain refreshment frequency) the context information and the recommended items, until the recommendation process is explicitly canceled by the user.

- *Explicit recommendations.* CARS are focused on providing items based on an explicit user query [CMBMRL+11, HMB13, BER14]. In a mobile environment, due to the limitations of mobile devices, entering a query manually may be inconvenient. Thus, in the last years, a key challenge is to recommend proactively relevant items without explicit requests from the mobile users [WHBGV11, VWB11, GBAH12, TZAQL12, GWH13, BRLW15, ABF16]. For this, context-aware recommendation systems should be able to automatically react to the user's contextual conditions in real time. For example, if the

current situation of the user is appropriate, then the system should recommend items of interest without the need of an explicit query.

In this thesis, we attempt to overcome the current deficiencies of CARS, and bridge the existing gap between CARS and mobile computing.

1.3 Overview of the Work

Our main contribution is a study of context-aware recommendation systems for mobile computing environments. First, we have proposed a generic and flexible architecture that supports context-aware recommendations in mobile environments over a distributed infrastructure of local databases, which are located on user's mobile devices and that contain information about items rated by users in specific contexts. In this distributed architecture, every mobile device acts as an independent node, which communicates with others to exchange information under specific conditions, using mobile P2P networks to facilitate the enrichment of the local databases of mobile users in a cost-effective way.

The proposed architecture supports pull-based context-aware recommendations, which are reactive recommendations obtained as an answer to a query submitted by the user. Besides, it provides push-based context-aware recommendations, where recommendations are automatically delivered to the user without explicit requests from her/him, when the current context is appropriate for it. In order to bridge the existing gap between context-aware recommendations and mobile computing, both recommendation approaches continuously re-evaluate the list of recommended items and automatically update the context information (e.g., using sensors of different types). The architecture is able to exploit not only static but also dynamic context information in mobile environments. Besides, no assumption is made about the movements of the objects involved in the recommendation process (users and items), that can be moving through predefined paths or freely in the open space. During the design of the architecture, we consider the peculiarities and generalities of different scenarios of CARS proposed in the literature, in order to obtain a framework that is sufficiently generic and flexible.

Furthermore, motivated by the lack of datasets available for the evaluation of CARS, we also developed a synthetic data generator for CARS, called DataGen-CARS. It is generic and allows flexible configurations to set up any type of realistic recommendation scenario, as well as the generation of coherent data about users, items, contexts, and ratings. Our experiments show the interest of a tool like this one, given the scarce availability of rich context-enhanced datasets that can be used for context-aware recommendation evaluation purposes.

Finally, we have performed a set of experiments that, among other aspects, show the benefits of incorporating context information in a recommendation process and the feasibility of using a mobile P2P architecture for context-aware recommendation systems. Due to the difficulty of evaluating context-aware dynamic recommendation approaches in a real-world environment, we also developed an application for the

simulation and evaluation of a recommendation scenario for mobile users. For this purpose, we considered a specific use case: museum guidance, where visitors want to appreciate a large number of works of art of interest in a limited time, in order to maximize the profit of their visiting time to the museum. The proposed recommendation system, in addition to continuously suggest relevant items to the mobile user, is able to sort the suggested items according to the interest, current context, and trajectory of the user. Besides, it supports mobile P2P ad hoc communications in order to disseminate, in an opportunistic way, information about ratings stored in the mobile devices of the users. The use case scenario is built from both real data and synthetic data created by using DataGenCARS. The application makes it possible to evaluate context-aware dynamic recommendation scenarios easily and at a minimal cost.

In the following, we describe in more detail the main contributions of our work. Firstly, we present the main ideas of the proposed architecture. Secondly, we provide an overview of the context-aware mobile recommendation approaches considered in our architecture. Thirdly, we show the main contributions of DataGenCARS. Finally, we summarize some aspects of an extensive experimental evaluation that we have performed, including the study of the use case for dynamic recommendations in a museum.

1.3.1 Context-Aware Mobile Recommendation Architecture

MOONRISE is a generic and flexible architecture that tries to facilitate the development of context-aware recommendation systems for mobile environments. The main features of our architecture are:

1. *It supports context-aware mobile recommendations based on pull and push approaches.* Pull-based recommendations follow a request-response approach in which the recommendations are provided to the mobile user upon her/his request, while a push-based approach proactively recommends items to the mobile user when the current situation seems appropriate, without explicit user requests. In a mobile environment, both recommendation approaches can re-evaluate the suggested items at a certain frequency (e.g., every five seconds), since the contextual information of the items and/or the users involved can change quickly.

2. *During the context-aware recommendation process, the static and dynamic context information is exploited.* On the one hand, what we call *static context* is characterized by attributes whose values do not change frequently over time (e.g., the age of the user, her/his job, etc.). On the other hand, the dynamic context (e.g., weather, location, transport way, etc.) represents changing information that is associated with a specific moment. The dynamic context plays an important role in context-aware recommendation approaches for mobile environments, since these approaches are capable of continuously adjusting themselves to the user's interest according to the current situation [BBG12].

3. *It is capable of operating over mobile P2P networks*, which provides the benefits of distributed systems (e.g., the recommendations can be distributed among all users, enhancing the scalability and improving the privacy of the users) [SP14]. Each mobile device contains a local database and behaves like a peer. The local databases of mobile users can be enriched through the exchange of information among them under specific conditions.

4. Besides continually suggesting relevant items to the mobile user, *the recommendation system is able to order the resulting items according to the user's interest, dynamic context and expected trajectories.*

5. *The architecture is flexible and generic.* For example, the users and/or items involved in the context-aware recommendation process may be moving through predefined paths (e.g., people in a car following a transportation network) or freely in the space (e.g., people in a park).

In Figure 1.1, we present an overview of the previous ideas, which represent the main features and motivations of our architecture. In general terms, the architecture is designed for mobile environments (e.g., users with mobile devices, moving through predefined paths or freely in the space) and where the information can be stored in a distributed way on the users' mobile devices. It is able to exploit data available outside the local knowledge base (obtained through data exchanges in a mobile P2P network) to benefit from external useful information, by potentially using several communication technologies (e.g., Bluetooth, Wireless Fidelity –Wi-Fi–, etc.). Besides, the proposed context-aware recommendation approaches consider both static and dynamic context information. In this architecture, a key goal is to enhance the existing context-aware recommendation approaches by considering the features of mobile computing. In this way, mobile users could minimize the effort of explicitly entering context information into the recommendation systems, in such a way that contextual elements are automatically captured, by using sensors of different types (see Section 2.1.2).

The proposed framework is based on a multilayered architecture. Specifically, we use a three-tier architecture in which the presentation, application processing, and data management are independent. Below, we briefly describe the three levels of our architecture:

- *View layer.* It reflects the main components of the user interface.

- *Logic layer.* It contains the main modules of the system, such as a sentiment analyzer, a pull-based recommendation module, a push-based recommendation module, a sensing engine, a user reliability analyzer, and a rating reliability analyzer.

- *Data layer.* It provides access to data relevant for context-aware recommendations in mobile environments, managing the details of the specific persistence approach used (relational database, files, etc.) in a transparent way, by using

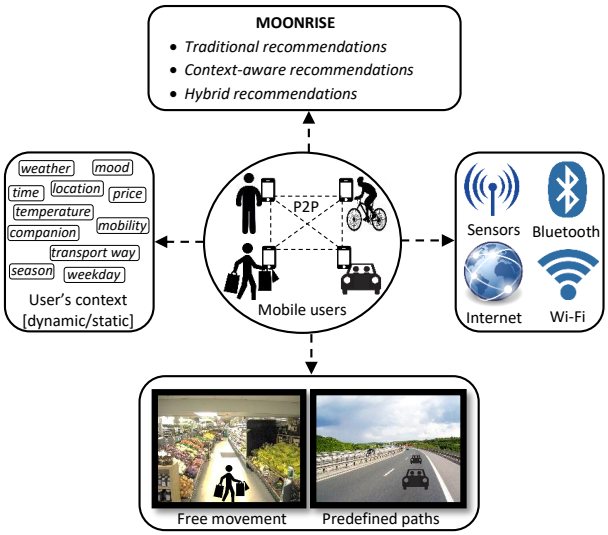


Figure 1.1: Overview of the main ideas of our proposal.

a repository manager, a user profile and context manager, and a data sharing manager.

In general, the architecture tries to facilitate the development of context-aware recommendation systems for mobile users. In Figure 1.2, we show a high-level overview of our architecture. In the view layer, the user can enter into the context-aware recommendation system the item type of interest as well as static context information, if the *Pull-Based Recommendation* module is applied in the system developed. As this module is designed for mobile environments, the dynamic context information is also exploited during the recommendation process, and can be obtained by using the *Sensing Engine* module, which uses the sensors embedded in mobile devices. In case of using the *Push-Based Recommendation* module in the context-aware recommendation system, the user will implicitly receive item suggestions when the user's context is appropriate, by using the available dynamic context information. Both context-aware recommendation modules access the local database of the mobile user for the generation of recommendations of items, by using the *Repository Manager* module. The provided recommendations are then updated in a continuous way (e.g., at a certain refreshment frequency). In Section 1.3.2, we present the basics of pull-based and push-based recommendations.

Besides, the *Data Sharing Manager* module allows implementing mobile context-aware recommendation systems by using a pure mobile P2P approach, where no centralized database or server exists. Instead, the mobile devices of the users propagate rating information in an opportunistic way, when they become neighbors from a communication point of view. In Section 3.2.3, we describe in detail the proposed data dissemination solution for these mobile P2P exchanges.

Further details about the architecture are explained in Chapter 3.

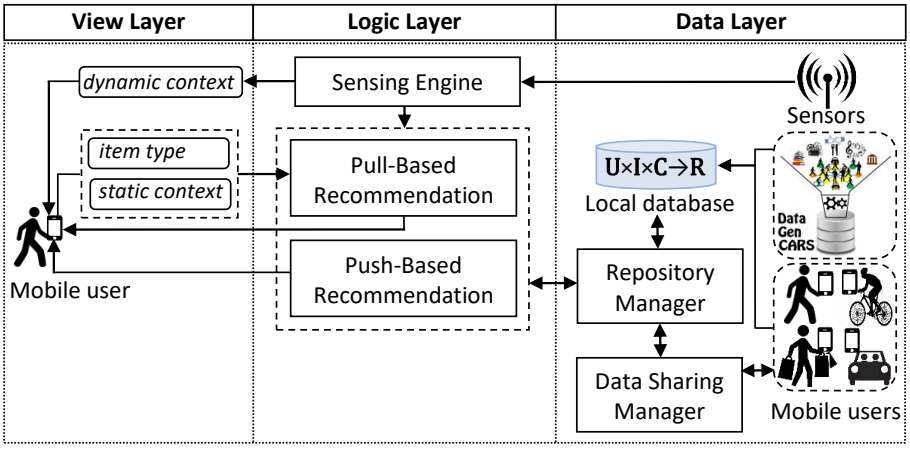


Figure 1.2: Overview of the proposed architecture.

1.3.2 Context-Aware Mobile Recommendation Approaches

In this section, we present the main modules of the *logic layer* of the architecture proposed in this thesis, which tackles both *pull-based recommendations* and *push-based recommendations*. We focus on the description of the main features of these approaches. In general, a key contribution of our work is that we take into account the features of mobile environments during the context-aware recommendation process. As specific and common benefits of both approaches, we highlight the following:

- *They are generic context-aware recommendation approaches for mobile scenarios*, where not only the static context is considered but also the dynamic context information.

- *They infer and update the dynamic context* (e.g., transport way, location, mobility, temperature, etc.) without user intervention, by using the *Sensing Engine* module, that exploits implicitly the information obtained from the available sensors.

- *They update the list of recommended items automatically* (e.g., at a certain refreshment frequency).

- *They allow the user to optionally specify her/his context criteria* (or preferences) about the importance of different context variables. For example, for a user who needs restaurant recommendations, the most relevant context variable could be the transport way she/he is using and/or the distance to each restaurant.

- *They support hard and soft constraints.* Soft constraints represent the preferences of the user regarding the impact of the different context variables, while hard constraints are specific conditions on the values of certain context variables that need to be satisfied.

In the following, we highlight the main aspects of these approaches. In Chapter 4, we present more details about the pull-based and push-based recommendation models.

1.3.2.1 Pull-Based Recommendations

In the architecture proposed, the pull-based recommendation module provides reactive recommendations, obtained as an answer to a query explicitly submitted by the mobile user and evaluated by the system as a continuous query [GU00, MXHA05]. During the pull-based recommendation process (see Figure 1.3), the user first introduces the *item type* (e.g., restaurant, movie, etc.) required (as the query). In addition, the user has the option to incorporate certain data about her/his *static context* (e.g., user preferences about item features, or basic personal data like her/his age or her/his genre) and *context constraints* (hard and/or soft constraints can be considered). Afterwards, a certain context-aware recommendation paradigm (*pre-filtering*, *post-filtering*, or *contextual modeling*) is applied to obtain appropriate recommendations for the current user. As the mobile environment can continuously change, the recommendation system must automatically update the *dynamic context* information (by using sensors) and evaluate the query in a continuous way. This continuous revaluation process will be performed until the user decides to cancel the query.

In Section 4.1, we explain the pull-based recommendation approach in more detail.

1.3.2.2 Push-Based Recommendations

Push-based approaches are usually exploited by context-aware recommendation systems that detect some condition in the environment that triggers the appropriate recommendation process. As an example, when it is time to have lunch and the user is outside in a foreign city, she/he could automatically receive appropriate recommendations of restaurants in the area.

As part of the logical layer of the architecture, we propose a generic push-based recommendation approach that supports proactive recommendations, provided without explicit requests from the mobile user. It is built on a multi-layer model which is general and can be adapted to different mobile computing scenarios and domains. Our proposal is based on the joint use of the pre-filtering and post-filtering paradigms, adapted for mobile environments by including a continuous revaluation of item recommendations and context updates. The push-based recommendation process follows the following phases (see Figure 1.4):

1. *Recommendation triggering*: phase that decides when the recommendation process should start.

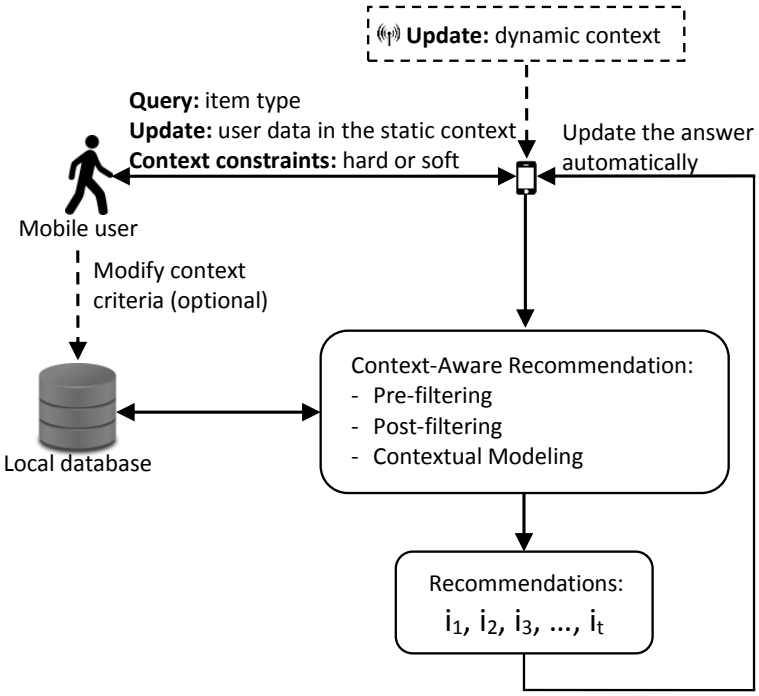


Figure 1.3: Overview of a pull-based recommendation scenario.

2. *Pre-filtering*: phase that filters items out of the scope of the user's context, and then a recommendation algorithm is used to obtain a set of possible items to recommend.
3. *Post-filtering*: phase that resolves conflicts between different types of items.
4. *Results display*: phase that presents the results to the user.

The push-based recommendation model proposed in our work is based on the definition of the concepts of *context* and *environment*, takes into account the impact of dynamic events, and includes all the actors that may play a role in a mobile context-aware recommendation process. In addition to the general benefits mentioned at the beginning of Section 1.3.2, this approach presents the following advantages:

- It recommends items of interest to the user without an explicit user request.
- It decides when it is appropriate to push recommendations to the user, avoiding an overload of information on the user's mobile device.
- It uses a generic contextual model based on several concepts: contexts, environments, agents, users, events, and activities. This model can be used in any recommendation domain.

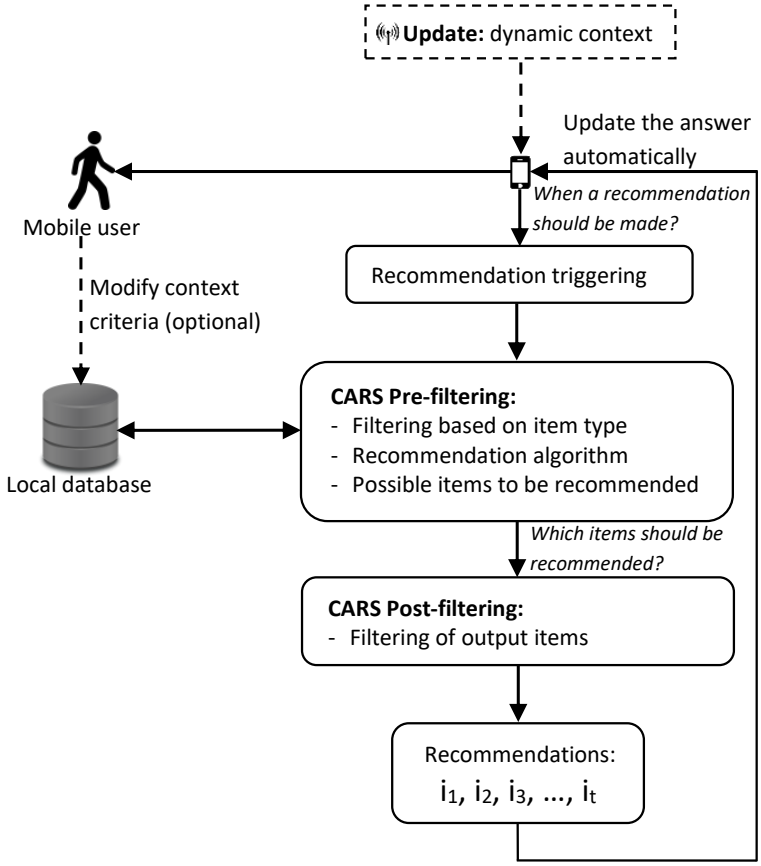


Figure 1.4: Overview of a push-based recommendation scenario.

- It allows the integration of different stakeholders and entities participating in a recommendation scenario. For example, a recommendation provider can set up environment managers that define the required conditions for a user to be a potential target of certain recommendations.

In Section 4.2, we explain in more detail the proposed design for push-based recommendations.

1.3.3 Generation of Synthetic Datasets for the Evaluation of CARS

Evaluating and comparing recommendation models is a key issue [SG11, HKTR04]. However, traditional well-known datasets are not suitable for the evaluation of CARS, due to several reasons:

- They do not usually incorporate contextual information that can be exploited.
- They are usually oriented to very specific domains, such as movies, music, or travel-related ratings and reviews.
- They are costly to collect in terms of the resources required (many mobile users actively using mobile applications to rate items in a wide area would be needed) and can usually lead to privacy problems, especially when trying to extract the contextual information.
- They are static, i.e., they capture data during a fixed period of time and under certain circumstances, but one cannot assume that the collected data will be representative enough for fully testing recommendation algorithms in any sort of scenario.

There are some datasets incorporating contextual information to test algorithms for CARS (e.g., STS [BERS13, EBRT13]). However, they are usually incomplete, since most users prefer not to disclose their context when they rate items, or they may just be lazy to do so. Moreover, these datasets do not cover all the possible potential scenarios that researchers would need to verify the validity and generality of their recommendation approaches.

Motivated by the reason indicated above, we developed DataGenCARS, a synthetic dataset generator that can be used for the automatic generation of datasets which are appropriate for the evaluation of context-aware recommendation algorithms. According to our own study (see Section 7.5), there are different tools to build synthetic data, but none of them is suitable to generate datasets for the evaluation of CARS. DataGenCARS is very generic and can fit different application domains and sets of needs, by appropriately defining a set of input data files that will direct its behavior. Figure 1.5 describes a simplified basic workflow of the dataset generator. However, the tool can also support other workflows, depending on the purpose of the dataset to be generated:

- Generation of a dataset similar to an existing one.
- Generation of a completely-synthetic dataset.
- Increasing of the number of ratings in an existing dataset.
- Filling of unknown context information in an existing dataset.
- Introduction of noise or unknown values in an existing dataset.
- Composition of different workflows (e.g., first a workflow that generates a synthetic dataset and then a workflow that increases the number of ratings).

DataGenCARS represents an interesting tool that can help to alleviate the problem of scarcity of datasets suitable for the evaluation of recommendation strategies, especially those that require rich context information. It presents features such as: a

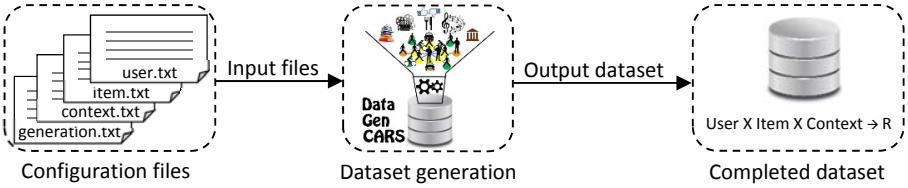


Figure 1.5: Simplified basic workflow of DataGenCARS.

flexible definition of user schemas, user profiles, types of items, and types of contexts; a realistic generation of ratings and attributes of items; the possibility to mix real and synthetic datasets; functionalities to analyze existing datasets as a basis for synthetic data generation; and support for the automatic mapping between item schemas and Java classes. In Chapter 5, we explain in detail the features of DataGenCARS.

1.3.4 Experimental Evaluation

In this thesis, we have performed an extensive experimental evaluation that supports our proposals and conclusions. For example, we can mention experiments that show that the incorporation of context information in the recommendation process increases the effectiveness of recommendation systems for mobile users, and others that show the feasibility of a mobile P2P approach. Due to the difficulty of evaluating context-aware recommendation systems for mobile users in the real world, we also used an example of a case study to simulate and evaluate a scenario of dynamic recommendations for visitors to a museum.

In the museum use case, a context-aware recommendation system is able to provide the visitor with accurate guided tours through a museum, taking into account different aspects of the context, such as opinions of other visitors, time constraints, her/his current location and trajectory, and her/his tastes, among others. We designed the system in a push-based manner, in such a way that recommendations are automatically provided and updated whenever it is considered relevant, without the explicit user's intervention. The exchange of opinions between the mobile devices of the visitors is opportunistic and relies on a mobile pure P2P architecture, which exploits short-range wireless ad hoc communications. Hence, there is no need of a fixed support infrastructure and no centralized server is devoted to collecting all the rating information provided by the users.

Moreover, in the use case, we use real data about the Museum of Modern Art (MoMA) in New York [The16], including information about paintings and sculptures as well as the map layout of the museum. Besides, we complete the real dataset (data about works of arts and map information) with other synthetic data (ratings provided by the users about the works of arts in specific contexts) needed to evaluate different CARS algorithms, by using DataGenCARS.

In Chapter 6, we explain in detail the use case scenario. Summing up, the novel contribution of this case study is twofold:

- We present a design and implementation of a recommendation system for museum visitors, which is able to exploit context-awareness for mobile devices. The system collects information from other users (i.e., their opinions about the works of art visited), by means of an information exchange service, and offers personalized recommendations about the next works of art to visit, taking into account context factors such as the location and trajectory of the user.

- We contribute to filling the gap between the design and evaluation of CARS, by exploiting the DataGenCARS tool in a mixed scenario built using both real and synthetic data. Up to our knowledge, this is the first experience of using a tool like this one to generate a mixed scenario for the evaluation of a recommendation approach for a real use-case scenario. This is particularly valuable because, especially for indoor environments, there is a lack of rich datasets that can be used to evaluate these kinds of systems.

There are also experiments that support the development of the DataGenCARS tool. All the experimental results are available in Chapter 8.

1.4 Structure of the Thesis

This thesis is composed of nine chapters, including this one, where we have introduced our motivation and main contributions. The remainder of the thesis is structured as follows.

In Chapter 2, we review the technological context of this work. In order to facilitate the understanding of this thesis, we provide an overview of general issues in mobile computing, traditional recommendation systems, the emergence of new recommendation techniques as a result of the incorporation of contextual information, and some considerations about the evaluation of recommendation systems.

In Chapter 3, we present our general and flexible context-aware recommendation architecture, that aims to facilitate the development of context-aware recommendation systems for mobile computing environments. Besides, we describe the main modules of each layer of the architecture.

In Chapter 4, we propose two approaches to the problem of context-aware recommendation for mobile environments. On the one hand, the pull-based recommendation approach provides reactive recommendations, obtained as an answer to a query explicitly submitted by the user, and evaluated by the system as a continuous query. On the other hand, in the push-based recommendation approach, the user would not need to explicitly request recommendations about specific types of items she/he is interested in. Instead, the system automatically provides suggestions (relevant items) to the users proactively.

In Chapter 5, we describe DataGenCARS, a complete Java-based synthetic dataset generator that can be used to obtain the required datasets for any type of scenario desired, allowing a high flexibility in the generation of appropriate data that can be used to evaluate context-aware recommendation systems.

In Chapter 6, we focus on a specific case study and describe the development of a prototype that facilitates the simulation and evaluation of the use case scenario of a museum, where the mobile visitors continuously receive context-aware recommendations.

In Chapter 7, we present an exhaustive survey of the state of the art on context-aware recommendation systems for mobile environments. We review some of the most relevant existing techniques for location-aware recommendation systems in several recommendation scenarios. In addition, we consider works related to mobile P2P recommendation systems. Moreover, we describe and compare different approaches existing in the literature to build synthetic datasets for evaluation purposes.

In Chapter 8, we present an experimental evaluation. Mainly, we focus on analyzing a set of experiments that support our proposal. Moreover, some experiments illustrate the interest and the benefits provided by DataGenCARS. We also evaluate context-aware dynamic recommendations in the specific use case of a museum visit, considering a decentralized P2P environment.

Finally, this thesis concludes in Chapter 9, by presenting our conclusions, main contributions, and some future research directions.

In addition, we provide three complementary appendices. Appendix A shows relevant classes of the architecture prototype. Appendix B presents some details of the DataGenCARS architecture. Finally, Appendix C discusses an evaluation of the proposed keyword-based item type searching approaches.

Chapter 2

Technological Context

In this chapter, we introduce in detail the technological context necessary to facilitate the understanding of the problem of context-aware recommendations in mobile environments. In Section 2.1, we focus on mobile computing, describing its current trends and related technologies. In Section 2.2, we provide an overview of traditional recommendation systems. Specifically, we describe the basics of the main types of recommendation systems, highlighting their differences, advantages, and limitations. In Section 2.3, we describe the concept of context-aware recommendation systems, as an extension of traditional recommendation techniques, as well as some challenges described in the literature. We focus on the explanation of the main paradigms for context-aware recommendation. Besides, we specify the importance of taking into account the features of mobile computing scenarios in the development of context-aware recommendation systems, where the user's context is highly dynamic. As a special case of context-aware recommendation systems, we also explain the specific case of location-based recommendation systems, where the contextual information exploited is basically the user's location. Finally, in Section 2.4 we discuss the key aspects to take into account during the evaluation of recommendation systems and several evaluation challenges to be addressed.

2.1 Mobile Computing

The emergence of portable devices (e.g., mobile phones, smartphones, portable computers, tablets, etc.) and advances in wireless networking technologies gave rise to a new paradigm of computing, called *mobile computing*. In mobile computing, users with portable devices have access to a shared infrastructure independent of their physical location [FZ94]. This provides flexible communication between people, as well as continuous access to data and network services *anywhere and at anytime*.

In Figure 2.1, we show an overview of a mobile computing scenario, where we can see that there are alternatives for long-range communications (e.g., 3G and 4G) [FFF+06, AA14], that require a wide-area infrastructure, and short-range com-

munications (e.g., Wi-Fi and Bluetooth) [LSS07, SK17]. A mobile environment infrastructure, represented in Figure 2.1, is composed by portable devices and base stations, which serve all the mobile devices within their coverage area or cell, by using wireless communications. The communication among base stations is wired. Thus, base stations allow the communication between mobile devices and the hosts of the fixed network. Moreover, mobile devices can directly interact without any supporting infrastructure through ad hoc P2P interactions, by using technologies such as Wi-Fi or Bluetooth. In public places (e.g., coffee shops, hotels, airports, libraries, schools, supermarkets, etc.), there are hotspots that offer internet access to the mobile devices, typically using Wi-Fi technology.

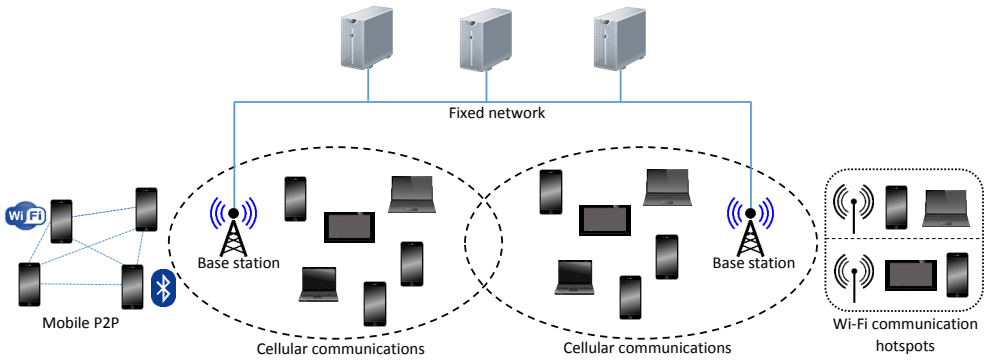


Figure 2.1: Overview of a mobile computing scenario.

In the last years, the use of mobile devices has increased. According to the statistics portal *statista*, the number of mobile phone users worldwide in 2017 is about 4.77 billions and is forecasted to increase to 5.07 billions in 2019 [sta17b]. In an analogous way, the number of smartphone users worldwide is about 2.32 billions in 2017, and is expected to increase to 2.71 billions in 2019 [sta17c]. By 2020, 80% of all mobile data traffic will come from smartphones [Eri15]. In 2021, there will be more people with smartphones than people with running water, according to the latest predictions revealed by Cisco [Dru17]. Moreover, according to [LMA+16], the explosive use of smartphones has led to a paradigm shift in many computing domains. Hence, the great interest and motivation of researchers in mobile computing is well motivated by the market expectations and existing trends.

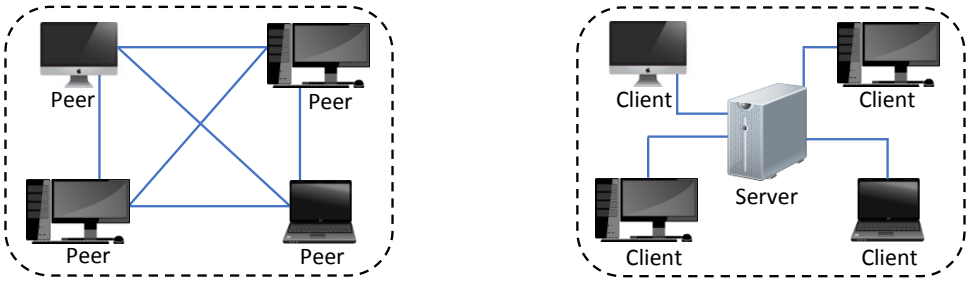
Mobile and wireless technology has experienced great progress in recent years. This combination has attracted a wide community of developers that are continually releasing new applications and services. The leading mobile application stores contained more than two million applications in 2016. For example, the number of available applications in Google Play [Goo12b] was about $2,200,000$ in June 2016 [sta17a].

In the rest of this section, we describe some technologies related to mobile computing. First, we introduce the basics of P2P networks and their integration with mobile computing (see Section 2.1.1). Second, we present examples of applications that use sensors of mobile devices in dynamic environments (see Section 2.1.2). Third, we

describe the main features of context-aware computing as a specific paradigm within the mobile computing environment (see Section 2.1.3).

2.1.1 P2P Networks

A P2P network is an alternative to a centralized (client-server) architecture, where there is usually a server and many clients (see Figure 2.2). Instead, in its purest form, a P2P network does not have the concept of centralized server: a node rather, all the computers are equally clients and servers at the same time, and so they are generally called peers or nodes. As a client, it can request resources from other nodes, and as a server, it can provide resources to other nodes.



(a) A P2P based network.

(b) A server based network.

Figure 2.2: P2P and centralized networks.

Oram [Ora01] defines P2P in the following way: "Peer-to-peer is a class of applications that takes advantage of resources –storage, cycles, content, human presence– available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS and have significant or total autonomy from central servers."

In other words, P2P networks allow exchanging resources (e.g., documents, images, songs, etc.) in a distributed way in a computer network. In the context of file-sharing, Napster (for music file sharing in MP3 format) [PF99, Fox01], Gnutella (network protocol, mainly used for file sharing) [FP00, Rip01, CRB+03, Mil04] and BitTorrent (communication protocol for file sharing) [GCX+07, YLZC17] are examples of popular P2P applications. The result to a keyword-based query is obtained through a collaboration of peers, where each peer contains a subset of the global results (e.g., text, audio, video, etc.) [NBM+06]. When a user submits a query from one of the nodes, it is sent to a set of nodes in the P2P network. Then, a list of results is obtained at each node individually, taking into account only the information that it stores. Finally, the combination of the lists sent by the nodes, is the final result of the initial query. An example of a collaborative P2P search is the Web search engine MINERVA, presented in [BMT+05].

In media streaming, there is great interest in using P2P networks [XHHB02, THD04, HC04a, YJC07, NMJ+13, HCXZ14, KG17]. Hence, a large number of P2P media streaming systems for audio/video sharing amongst multiple interconnected peers have been developed [JDXB03, HHB+03, Li04, KN10, PFS17, DYS+17]. The studies presented in [LGL08, GCM11] provide an overview on existing P2P video streaming applications in the related literature, including aspects of P2P streaming design, security, and privacy.

P2P networks have also been applied to traditional recommendation systems as a solution to the problems of scalability and calculation complexity, due to the rapid increase of information in centralized databases used by these systems. Hence, several methods and frameworks to build P2P recommendation systems have been proposed in the literature [HXYS04, PM06, WPLR06, KKC08, MBDR10, DPPV11, dCPHSS12, SP14, KRR17].

In comparison to the traditional centralized model, P2P networks provide certain advantages that justify their usage in building distributed systems [HAY+05]. For example, nodes provide resources and use resources of others in a collaborative way through the network (thus maximizing the effective use of the resources available), they will continue to operate even if one of the nodes is disconnected, improve scalability by avoiding the dependency on centralized servers, eliminate the need to use a costly hardware infrastructure by enabling direct communication among clients, and reduce bandwidth costs. However, this type of architecture presents some challenging disadvantages [PBV05], such as weak security, lack of global control of access to resources (as there is no server), the backups have to be done on each peer individually, etc.

According to the degree of centralization, P2P networks are classified in several types [LCC+02, PBV05]:

- *Centralized*. It performs resource transactions through a server, which serves as a connection point between two nodes. The server stores a centralized index with meta-information of the resources stored locally on the nodes of the network community. In this type of architecture, during the search process the nodes in the P2P network apply the query to the existing index on the central server to find the location of the nodes that contain the desired resources. For example, the Napster application applies a centralized indexing.

- *Pure or decentralized*. It is a distributed system that does not require any type of centralized service. In such pure P2P networks, all the nodes act equally as servers and clients. Currently, it is the most common and popular P2P architecture. Gnutella is an example of a traditional application that applies a pure P2P network.

- *Hybrid or mixed*. It combines the centralized and pure P2P networks. In this case, there is a central server that manages the resources related to the use of bandwidth, routing and communication between nodes without the need to know the identity of each node. The main advantage of this type of network

is that it can incorporate more than one server for the management of shared resources. If the server (or the servers) fail, then the group of nodes can perform a direct communication between them. Thus, it is possible to continue sharing and downloading resources without servers. Among the most popular examples of protocols using a hybrid architecture, we can highlight BitTorrent.

2.1.1.1 Mobile P2P Networks

With advances in wireless technology, P2P networks have played an important role in mobile computing [RM06, SYBA10, ZI16]. The decentralization of information has enabled large-scale dynamic interactions between mobile devices, which can quickly exchange data using short-range wireless communication.

In this context, a new domain of P2P applications for Mobile Ad Hoc Networks (MANETs) [CCL13, BCGS13, CG14] have appeared. For example, we can mention frameworks such as *Proem* and *Peer2Me*, which facilitate the development of P2P applications in MANETs [KSP+01, WBS07]. Another example is the *COMPASS* protocol, that dynamically determines the data access paths that minimize the overall latency in heterogeneous mobile Chord-based P2P systems [SPS13]. *iTrust* [LMMSC14] is a P2P retrieval system over Wi-Fi Direct [CMGSS13] for MANETs. In the same line, mobile P2P video/audio streaming systems is another example [IBHD16]. A survey of P2P content/file sharing systems for MANETs is provided in [SAQM17].

In the field of recommendation systems, there are also specific proposals. For example, a P2P recommendation system that suggests products and services to mobile customers is proposed in [Tve01]. Another example, is the *iTravel* system, that recommends attractions to tourists in a mobile P2P environment [YH13]. A context-aware recommendation system that proactively pushes news in a mobile P2P network is presented in [YY10].

Moreover, the increased interest of mobile P2P networks and Intelligent Transportation Systems [DD10, ZWW+11], has led to intensive research on Vehicular Ad Hoc Networks (VANETs) [WL09, CMS15, RGAQ17], as a specific type of MANET. The combination of these technologies facilitates the rapid exchange of relevant information with other cars (e.g., accidents, available parking spaces, obstacles in the road, real-time traffic information, etc.) in a P2P way. In [IDTL15], the authors provide a detailed overview of data management techniques for vehicular networks.

2.1.2 Sensors

A sensor is a device that converts a physical phenomenon of the environment into an electrical signal [Wil04, JGT+05]. According to the way the data is captured, sensors can be classified into the following types [IWD14, IHTLdCRH15]:

- *Physical or hardware sensors:* they provide certain raw data captured from the environment.
- *Virtual or software sensors:* they provide higher-level observations usually obtained by fusing the measurements of several sensors (e.g., a location obtained

by combining different positioning mechanisms) [KPJ06].

- *Social sensors:* they provide data based from the social media, such as data posted in social networks (e.g., Facebook, Foursquare, and Flickr), blogs, or microblogs (e.g., Twitter) [RMZ13]; as an example, the proposal in [AMO13] exploits microblogs to detect events in the vicinity.

- *Human sensors:* humans can also provide interesting data using their own senses or managing other sensors in specific ways; so, they can provide *volunteered geographic information* (VGI) [Goo07] or participate in *spatial crowdsourcing* [KS12] tasks.

Users with their mobile devices have become an important source of sensor data, as it is possible to exploit the sensors available in existing smartphones [CEL+06, GYL11, LML+10]: inertial sensors, compasses, GPS receivers, microphones, cameras, proximity sensors, ambient light sensors, accelerometers, gyroscopes, temperature sensors, pressure sensors, and so forth. These sensors have facilitated the development of more flexible and dynamics systems in several domains, such as healthcare [CMT+08], social networks [MLF+08], environment monitoring [MRS+09], and transportation [TRL+09, Fle13].

In recent years, the use of sensors is an essential element for context detection [IHTLdCRH15]. Below we mention, for each type of context, the mechanisms or sensors typically used to capture the contextual information indicated [VMO+12, Asa13b]:

- *Computing context.* It includes aspects such as the processing power of the CPU, amount of memory, current CPU and memory use, battery level, etc. It is captured implicitly by the device itself.

- *User context.* It refers to user's interests (or goals) and can be obtained explicitly, for example, through a user registration process [SGRB08] or by using modules able to capture explicit interest indicators (e.g., the system identifies thematic groups by analyzing social annotations of each user's preferred resources) [SBMD10]. Implicit approaches obtain the user's context information through interactions of the user with the system [ZR08].

- *Location context.* It refers to the spatial location (e.g., latitude and longitude) of a person or object. In *outdoor scenarios*, it is often sensed by using positioning mechanisms (e.g., GPS) [BHE00, BZM12, BKR13, ZJL+16, TGG+17], while in *indoor scenarios* the positioning technologies commonly used are based on short-range signals (e.g., Bluetooh, Wi-Fi and infrared), or by using ZIP code [HNV06, DMG12], trajectory data [TS05, TS06a, CSdVR11], and explicit methods that require scanning Radio Frequency Identification (RFID) tags [EBOY07, MPS14, DP16, XZYN16, HPC+16], among others.

- *Social situation context.* It refers to relations between users (e.g., family member, friends, neighbors, co-workers, etc.). For example, this can be information about whether a user is with her/his manager or with a co-worker. The

social situation can be explicitly captured from a manual representation of the group structure [PM08], or implicitly by capturing data from the system (e.g., enrollment data from learning management systems [CC08] or social networks [LPCY17]). In order to obtain indications of the level of collaboration between different member of a group, there are systems that infer the social relations by analyzing interactions between users [HSG10].

- *Physical context.* It is typically acquired from the environment implicitly (e.g., with a thermometer sensor to determine the temperature of the environment, a light sensor to know if it is day or night, a microphone sensor to measure the noise level, etc.) [BDR07], or captured explicitly by the user [CB05].

- *Time context.* It can be entered either explicitly by the user (e.g., available study time [CB05]) or determined implicitly by checking the device's internal clock.

- *Activity context.* It is often achieved through mobile phone sensors (e.g., accelerometer, gravity sensor, magnetometer, microphone, and gyroscope), without interfering with the user's lifestyle. Some systems require explicit user interactions, such as scanning a QR (Quick Response) code [TCL11] or providing manual text input [CC08], to obtain activity context information.

In Section 2.1.3, we revisit some concepts of context under perspective of context-aware computing.

2.1.3 Context-Aware Computing

The interest of exploiting contextual information gave rise to the emergence of context-aware computing as a paradigm within mobile computing [CK00, MN13]. Several perspectives on how mobile applications consider the context have been presented in the literature [SAW94, CBJC11, FCF+14, BN16, ALLR16, CLLP17]. In general, the main goal of context-aware applications is to examine the user's context and react to the changes of the dynamic environment to discover information of interest [HS09].

In [ADB+99], the authors define the *context* as "any information that can be used to characterize the situation of an entity", where an entity could be "a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Other definitions of context have been introduced in the literature related to the context-aware computing field (e.g., [Dey01, Dou04]). The meaning of *context-aware* was defined in [ADB+99] by indicating that "a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task".

Examples of elements defining the context could be the location, temperature, weather, noise level, activity, traffic conditions, lighting, time of day, week, season of the year, network connectivity, nearby resources, communication bandwidth, and people accompanying the user, among others. There are certain types of context elements that, according to the circumstances, could be more important than others;

for example, if it is raining a person could prefer to stay at home watching a movie rather than to go to run (i.e., the weather element in this case is more important than others). Sometimes authors classify the context by categories. For example, the following categories are used occasionally [Asa13b]:

- *Computing context.* It describes hardware (e.g., storage or CPU capabilities), software (e.g., operating system), or network characteristics (e.g., network connectivity, communication costs, and communication bandwidth) of the mobile device and nearby resources.
- *User context.* It describes the user environment, including the location of the user, social situation, and people nearby.
- *Physical context.* It describes the environmental situations related with the user or system. For example, the amount of lighting, traffic conditions, temperature, weather, and noise levels.

There are also authors who include other categories [CDC14], such as *date* and *time* information (e.g., hour, week, month, semester, year, time available for a user's activity, etc.), or the *user's activity*, that includes characteristics such as tasks, objectives and actions executed [BBC+08, DTB12]. In addition, it can refer to the current activity that the user is performing (e.g., walking, running, going upstairs, riding an elevator, watching TV, driving a car, riding a bike or bus, etc.). From the perspective of the source and the persistence of information, the context can be divided into two main types [HIR02, Pas05, AT11], which are:

- *Static context.* It does not change its value frequently (e.g., address book, contact list, user profile, user preferences, hardware profile, etc.).
- *Dynamic context.* It is highly variable. Examples of dynamic context are the user's location, user's current task, vicinity to other people or objects, weather, temperature, speed, time, system status, user's emotions, etc.

Context awareness represents a generalized model of relevant data input (both implicit and explicit) that allows an application to react to its environment. According to Adomavicius and Tuzhilin [AT11], the contextual information can be acquired in several ways, such as the following:

- *Explicit acquisition*: when the user enters contextual information directly into the system (through input fields of the system, by asking the user to fill out a form or answer specific questions, etc.).
- *Implicit acquisition*: when the context is obtained by observing the user's behavior, relevant data, or the environment (e.g., the user's location detected by the mobile device).
- *Inferred acquisition*: when the system obtains context data using statistical or data mining methods.

Generally, the computing context is acquired implicitly by embedded sensors in mobile devices (see Section 2.1.2). In [BDR07, PZCG14, YLS+16], the authors provide a survey on context-aware systems, highlighting the different types of sensors used.

2.2 Traditional Recommendation Systems

A *Recommendation System* (RS) is an application which suggests relevant items (articles, products, objects, or places) to users [RV97, RRSK11, LMY+12, BOHG13, LRU14]. It tries to adapt its proposals to each user individually, based on her/his preferences. These recommendations can be seen as an advice about relevant items that are considered of interest for a particular user. For example, in a scenario of books the recommendations should be books that are expected to be relevant for the user (and so they should be read before others), in a scenario of travel destinations the recommendations would be places that according to the user preferences will be more attractive for the user, in the context of a digital newspaper the recommendations would be the news that the user could find interesting, in the context of movies the recommendations would be movies that the user would probably like, etc. More formally, the task of recommendation can be formulated as indicated in Definition 1 [AT05].

Definition 1 *Let $U = \{u_1, u_2, \ldots, u_k\}$ be the set of users and let $I = \{i_1, i_2, \ldots, i_t\}$ be the set of all possible items that can be recommended. Let $f : U \times I \rightarrow R$ be a utility function that measures how useful item i is for user u, where R is a totally ordered set (e.g., non-negative integers or real numbers within a certain range). Then, for each $u \in U$ the goal of a recommendation system is to find the item $i^*_u \in I$, not yet known to the user, that maximizes the utility function:*

$$i^*_u = argmax f(u, i)_{i \in I}$$

In Figure 2.3, we show the main elements of a recommendation system:

- The *input data* (e.g., the item type requested to the RS and information related to the user profile) are entered (explicitly or implicitly) by the user to initialize the recommendation process.
- A *database* stores information about user and item profiles.
- The *recommendation algorithm* uses the input data and the database to suggest a list of items to the user (also known as target user, current user, or active user).

On the one hand, *user profiles* have information about the characteristics (e.g., age, sex, occupation, country, etc.) and preferences (e.g., opinion on a rating scale about an item seen, purchased or visited) of the users. This profile information can be explicitly or implicitly entered by each user in the system. In the explicit case, for

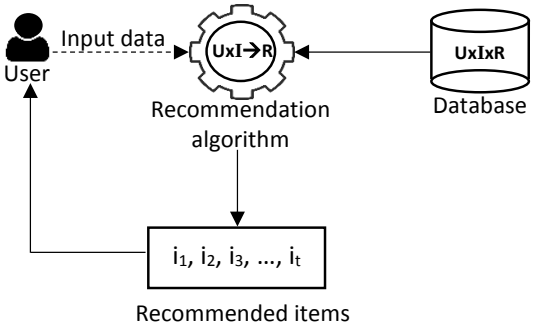


Figure 2.3: Overview of a recommendation process.

example, the user is asked to manually provide some profile information (e.g., Netflix asks the user to select some movies or series that might like her/him). In the implicit case, for example, the preferences are obtained directly from the user's interaction with the system, without requiring their intervention in an express way. On the other hand, *item profiles* contain the products or places (e.g., books, movies, restaurants, etc.) to recommend, which are typically characterized by features (e.g., obtained from a catalog of products, or provided by business owners, etc.), textual descriptions (e.g., extracted from external sources such as forums), and tags (e.g., generated by a user community), among other types of information (e.g., acquired from logs). In Figure 2.4, we show an example of basic information about the relationships between users and items in a movie recommendation system, in this example, only binary ratings are shown (like or not like).

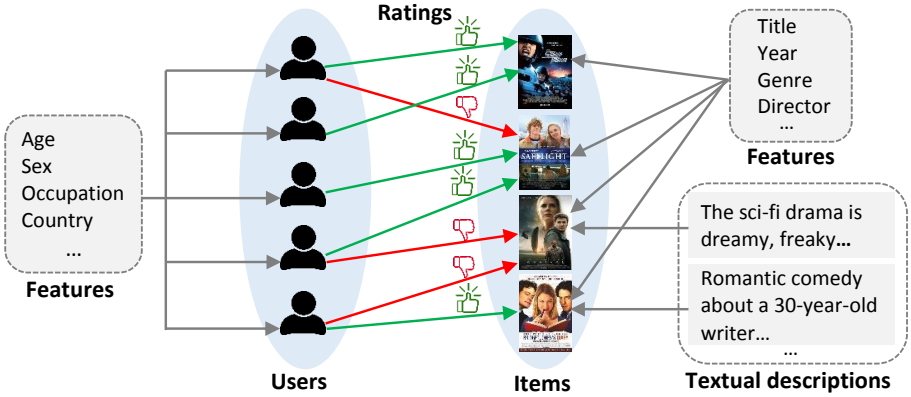


Figure 2.4: Example of user and item profiles in a movie recommendation system.

One of the fundamental tasks of a recommendation system is thus the prediction of a rating: for a particular item not seen by the user, the system should be able to estimate how the user would evaluate it. Then, if the predicted rating is above a

predefined threshold, then the item can be recommended to the user. A list of suitable items to be recommended to a target user is usually sorted according to the ratings predicted by the system. Depending on how the recommendations are obtained, a recommendation system can be classified in one of three categories [AT05]:

- *Collaborative filtering recommendation.* In user-based collaborative filtering, the user is recommended items that people with similar tastes and preferences liked in the past. In item-based collaborative filtering, is the same idea, but use similarity between items instead of users.
- *Content-based recommendation.* The user is recommended items similar to the ones the user preferred in the past.
- *Hybrid recommendation.* It combines collaborative filtering and content-based methods.

In the last decade of the $20th$ century, the use of these systems has increased in different application scenarios [LWM+15]. Most major companies use RS within their services: for example, we can cite Google [HCK05, DDGR07, DLL+10], Twitter [Dor07, KLZ12], eBay [Omi95, SKR01], Facebook [ZSHM07, NGL11, SRF13], LinkedIn [HBG+02, LGS+11b, SKS13, DVR17], NetFlix [HR97, Ama13, GUH16, Aro16], Amazon [Bez94, LSY03, Aro16, SL17], Spotify [Sta08, GC13, JMN+16], Yahoo! [YF94, KDK11], and Pandora [KGW00]. Recommendation systems have been proposed to recommend a whole range of items, including books, music, movies, news, touristic destinations, friends in social networks, and others [CLG+10, RRSK11, PKCK12, BOHG13, Eth14, KS16, ZCZ+17]. They are particularly popular in e-commerce [SKKR99, SKR01, WHF07, Lin14], as providing relevant recommendations to customers can help to improve their satisfaction and increase product sales. Given the continuous increase of the volume of information to which the users are exposed, recommendation systems are a very useful tool, able to learn the behavior of users and discover their preferences.

In general, the use of recommendation systems has been successful for solving the problem of information overload, the growth of the number of items sold, the sale of the most diverse and novel items, as well as to facilitate a better understanding of the user's needs and to increase the satisfaction and fidelity of the users [AT05, RRSK11]. However, there are still challenges and constraints that offer research opportunities [RRSK11, KS16], related to topics such as:

- The incorporation of *contextual information* during the recommendation process [AT05, AT11, LMCX13, Asa13b, PTG14b, AZK15].
- The *scalability* of recommendation algorithms, taking into account large real-world datasets [SKKR02, SKR05, TPNT09, YAG17, LCF+17].
- The support of *multi-criteria ratings* [AT05, LT07, ZHW+10, JKG12, AK15]. For example, a restaurant can be evaluated (e.g., on a scale of one to five) regarding different aspects or criteria (e.g., $rating\_food = 3$, $rating\_decoration = 4$,

rating_service = 5, and *rating_price* = 4), rather than in a single criterion rating (e.g., *rating* = 4) like in traditional recommendation systems.

- The *privacy-protection* between users in RS [SFR06, ZHW+10, MKS11, BP12, PGPM17, DGK17]. Recommendation systems must be able to keep the personal information of the users private, including their preferences, as users should not be tracked against their will [ABFO08].
- The design of recommendation systems that operate on *mobile devices* [TS04, KMtH06, AjKH06, PKK06, LP07, ARN08, SPGR08, GKMP14, RCK+17].
- The *proactive recommendation* of items without the need to generate explicit queries [SUPOK08, MBR09, YY10, SKP+14, ASM16, Sab16].
- The *diversity* of items recommended to a target user [ZH08, ZKL+10, VC11, HZ11, AK12].
- The *serendipity* (or novelty and unexpectedness of items) in recommendation systems [IGL+08, ZSQJ12, KWV16, MTAS17].
- The application of strategies that deal with the *sparsity problem* (the number of ratings provided by users is very small compared to the number of ratings unknown, and consequently the rating matrix is usually very sparse) [HCZ04, PPK05, YK08, KBV09, PHJ15, NKG15].
- The use of *distributed architectures* (e.g., P2P networks) in recommendation systems [MBDR10, MCR11, dCPHSS12, YH13, SP14, KM16]. For more details see Section 7.4.
- Recommendations to *groups of users* with common interests [OCKR01, Jam04, JS07, RGJDSRDA09, KRS13b, ZWF13, PDM17, FC17].

In Sections 2.2.1 and 2.2.2, we will address the fundamental characteristics of some of the types of RS previously mentioned.

2.2.1 Collaborative Filtering Recommendations

Collaborative filtering (CF) is the process of filtering (or evaluating) information using techniques involving the collaboration among several users, through their provided ratings [SFHS07, SK09, TH01, ERK11, SLH14]. The motivation of this idea is based on the usual decision-making process of people, where the opinion of known people (e.g., friends, family, experts, etc.) can greatly influence the decision made by a person. This is because people tend to trust the opinions of other people who generally have similar tastes. CF essentially automates the process of "word of mouth" recommendations, where people suggest products or services to each other [SM95].

These recommendation systems have been successful in research projects, such as *GroupLens* [RIS+94, KMM+97] for news, *MovieLens* [DKH+98] for movies, *Video Recommender* [HSRF95] for movies, *Ringo* [SM95] for music, and *Jester* [GRGP01]

for jokes. From a commercial perspective, CF recommendation methods are notably deployed into commercial websites, such as Amazon.com, Netflix.com, CDNow.com, Launch.com and MovieFinder.com.

The information domain for CF recommendation systems consists of *users* who have expressed preferences for several *items*. A preference expressed by a user for an item is called a *rating* and is frequently represented as a ⟨User, Item, Rating⟩ triplet. These ratings can take different ranges of values and forms. For example, some recommendation systems use rating scales of one to five stars, or integer or real values in the range of one to 10, while others use binary scales (e.g., like or dislike) or unary ratings (e.g., "has purchased", "has been", or "has visited").

The information of the user-item ratings can be represented as a matrix $User \times Item$ with ratings r. An example of a matrix of ratings can be seen in Table 2.1. Cells with question marks represent unknown rating values, which must be determined for the recommendation of items.

User/Item	Movie A	Movie B	Movie C	Movie D	Movie E	Movie F
User 1	4	2	3	?	?	4
User 2	5	3	?	?	1	?
User 3	4	4	1	3	?	?
User 4	3	3	?	4	1	3

Table 2.1: Example of a matrix of ratings.

The main tasks of CF recommendation approaches are to establish the similarity among users (user-based or user-user CF) or items (item-based or item-item CF), the selection of neighbors with similar tastes (user-based CF) or the selection of items with similar ratings (item-based CF), and the prediction of the items to be recommended. Depending on the specific algorithm used, collaborative filtering methods can be classified into the following categories [BHK98, SLG11]:

- *Memory-based collaborative filtering.* It is one of the most popular collaborative recommendation techniques, and it is based on algorithms to find the nearest neighbors (NN). For the prediction of new items, this technique analyzes the entire $User \times Item$ matrix of ratings to identify users or items with patterns of similar ratings.

- *Model-based collaborative filtering.* It applies machine learning techniques (e.g., Bayesian networks, linear classifiers, clustering, neural networks, association rules, etc.) to learn a model (or common patterns of behavior), by using available interaction information provided by the users to the system (i.e., the ratings provided by users). The model learnt is then used to generate the predictions about the missing interactions.

2.2.1.1 Memory-Based Collaborative Filtering

Memory-based collaborative filtering methods perform predictions by analyzing similarities between users or items [BHK98]. Below we briefly explain the user-based and item-based approaches:

- On the one hand, the *user-based collaborative filtering approach* (or user-user collaborative filtering) [SLH09, ZS10] identifies the k-nearest neighbors to the current user, and based on these neighbors it determines the prediction of the items not seen by the current user. The prediction $\hat{r}(u_a, i)$ of an item not seen can be calculated as the linear combination of the ratings $r(u_v, i)$ assigned to the item i by the users u_v most similar to u_a (see Equation 2.1).

$$\hat{r}(u_a, i) = C \sum_{u_v \in U^k_{u_a}} sim(u_a, u_v) \times r(u_v, i) \tag{2.1}$$

  where $C = \dfrac{1}{\sum\limits_{u_v \in U^k_{u_a}} \mid sim(u_a, u_v) \mid}$

  In order to give greater weight (in the prediction) to the ratings of the users that are more similar to the current user u_a, the similarity $sim(u_a, u_v)$ is included in the linear combination. Similar users are called neighbors and, for reasons of computational cost and noise elimination, the list of neighbors used in the prediction $\hat{r}(u_a, i)$ is truncated to the k nearest to u_a. Hence, $U^k_{u_a}$ represents the set of the k users more similar to u_a, and C is a normalization constant so that the resulting prediction $\hat{r}(u_a, i)$ remains in the same range of $r(u_v, i)$. Among others metrics, the similarity between the users can be calculated with the *cosine measure* or the *Pearson correlation coefficient* (see Equations 2.9 and 2.10, respectively).

  One problem with the prediction of ratings is that it does not take into account the different biases that users tend to have when evaluating items (e.g., on a scale of one to five, whereas for a user one $rating = 3$ means that the item is acceptable, for another user the same rating can be used for items that she/he does not like) [SLG11]. In order to soften this problem, Equation 2.1 is frequently normalized by including the average of the votes assigned by the user $\bar{r}(u_v)$, as shown in Equation 2.2.

$$\hat{r}(u_a, i) = \bar{r}(u_a) + C \sum_{u_v \in U^k_{u_a}} sim(u_a, u_v) \times (r(u_v, i) - \bar{r}(u_v)) \tag{2.2}$$

- On the other hand, the *item-based collaborative filtering approach* (or item-item collaborative filtering) [SKKR01, LSY03] takes into account the set of items rated by the current user to determine the k items more similar to an item that the user has not rated yet. Formally, the prediction $\hat{r}(u_a, i)$ for a target

item i is determined as the linear combination of the items rated by the current user $r(u_a, i_j)$, weighted by the similarity $sim(i, i_j)$ between these items and the target item (see Equation 2.3).

$$\hat{r}(u_a, i) = C \sum_{i_j \in I_{u_a}} sim(i, i_j) \times r(u_a, i_j) \tag{2.3}$$

$$C = \frac{1}{\sum_{i_j \in I_{u_a}} \mid sim(i, i_j) \mid}$$

As in the case of user-based collaborative filtering, the similarity between the items can be calculated using the *cosine* and *Pearson* measures (see Equations 2.9 and 2.10, respectively), exchanging users for items.

2.2.1.2 Model-Based Collaborative Filtering

Model-based collaborative filtering methods build a model that identifies patterns of common behavior, which is then used to perform rating predictions [Agg16]. Many of these methods are inspired by Machine Learning (ML) techniques. For example, the problem can be treated as a classification problem: the rating matrix can be viewed as a knowledge base, where the target classes could be the ratings and the features would be the rated items. Examples of algorithms within the ML category used in model-based collaborative filtering methods include Bayesian networks [BHK98, MP00], linear classifiers [ZI02], clustering [UF98, KM99], neural networks [BP98], and association rules [SKKR00].

In comparison to memory-based approaches, model-based CF methods are typically faster at query time, but they are time-consuming during the learning or updating phases [XLY+05, LM05]. In [PHLG00, GYT09, AT05], the authors combined the memory-based and model-based approaches, and empirically demonstrated that the use of this combined approach can provide better recommendation than traditional collaborative filtering approaches.

2.2.1.3 An Example of a Popular Model-Based Collaborative Filtering Approach: SVD

In terms of scalability and precision, matrix factorization techniques have gained popularity [KBV09, YHSD14, SZ17]. Among the existing techniques, Singular Value Decomposition (SVD) is one of the most used in the field of recommendation systems [Bra03, SKKR02], generating recommendations of high quality in a short time. SVD is based on the linear algebra theorem presented in Definition 2 [GR70]:

Definition 2 *A rectangular matrix A $(m \times n)$ can be decomposed into the product of three matrices: an orthogonal matrix U $(m \times m)$, a diagonal matrix S $(m \times n)$, and*

the transpose of an orthogonal matrix V $(n \times n)$, being represented as follows:

$$A_{mn} = U_{mm} S_{mn} V_{nn}^T$$

From the perspective of applying SVD as a recommendation algorithm, the rating matrix can be considered the rectangular matrix A, where m represents the number of users (located in the columns) and n the number of items (in the rows). Taking into account the previous definition, the three components (or matrices) resulting from the matrix decomposition would be the following: $U[m \times m]$, related with users, $S[m \times n]$, which is associated with items, and the diagonal matrix $V[n \times n]$, composed by singular values (sorted by decreasing importance).

The original rating matrix A can be approximated by considering the most important singular values of matrix S. The k values of the matrix S are thus a compressed representation of the data, and model the different factors considered in the recommendation. The most appropriate value of k depends on the data set.

SVD is a powerful technique when a large amount of data is available and we need to reduce the dimensionality [RRSK11]. The main advantage of the SVD method is its utility in cases where two users are not very similar despite having information from both. This is due to the ability of this method to relate users even if they have not rated common items.

2.2.1.4 Advantages and Disadvantages of Collaborative Filtering

In contrast to content-based recommendation (explained in Section 2.2.2), collaborative filtering has the advantage that it does not need information about the features of the items. In addition, collaborative filtering takes into account the global criteria of the user community in the estimation of votes, which provides an important advantage, enabling the possibility to recommend novel items (very different from those already seen) to users.

However, a collaborative filtering approach is not capable of generating good recommendations under certain circumstances, such as the following [SLG11, RRSK11, CCFF11]:

- The *cold start problem*, which occurs when a user or item is new in the recommendation system [Ahn08, BOHB12, LKH14b, HCC+16, HBOG17]. In the first case, for new users the system does not have information about their preferences in order to suggest suitable recommendations. For example, in the case of new items, since no users registered in the system have rated those new items yet, they cannot be recommended to any user. The emergence of hybrid approaches is one of the proposed solutions to solve this problem [SPUP02, PC09, Son16]. In Section 2.2.3, we describe hybrid recommendation systems in more detail.

- The *high computational cost* of the recommendation algorithm [PRPT05, GM05, SBM12]. This is due to the high dimensionality of the data produced by item-user interactions (e.g., ratings) and the number of comparisons needed to find the nearest neighbors. A potential solution is to perform comparisons offline, but

once new data are introduced in the recommendation systems, the comparison process must be repeated. Another way to attack this problem is to use parallel computing techniques [ZWSP08].

In Section 2.2, we mentioned some challenges of traditional recommendation systems.

2.2.2 Content-Based Recommendations

The general idea of the *content-based recommendation* model is to recommend items similar to the ones the user liked in the past, taking into account the description of the items (i.e., their features) [PB07, LGS11a]. This model needs, in addition to the rating matrix, the description of the items to recommend. For example, a movie could be described by its genre, director, actor, etc.

In content-based recommendation systems, the user and item profiles are considered. On the one hand, the *user profile* represents the preferences of the users according to the description of the items for which the user has shown interest. On the other hand, the *item profile* of a user is represented as a vector (or set) of k features that characterize the item i. Each dimension of the vector has associated a weight, which can be determined in several ways. One of the simplest methods is the binary representation (used in Table 2.2), where 1 represents the presence of a feature and 0 its absence.

Item	Comedy	Violence	Action	Drama	Romantic	**Rating**
Movie A	0	0	1	1	1	4
Movie B	0	0	0	1	1	5
Movie C	0	1	1	0	0	2

Table 2.2: Example of item profiles of movies.

Taking into account both profiles (user and item profiles), a content-based recommendation model suggests the items that have a high degree of similarity to the user's preferences. For example, if Table 2.2 represents movies watched by a particular user, we can conclude that the user B prefers movies of *drama* and *romantic* genres.

A content-based recommendation approach uses measures to determine the similarity between items. Among the different existing measures, the *Euclidean distance* (see Equation 2.7), the *cosine of the angle* (see Equation 2.9) and the *Pearson correlation* (see Equation 2.10) are frequently used. For more details about similarity measures, see Section 2.2.4.

2.2.2.1 From Text to Structured Features

Sometimes, the information about the items is a textual description (or a document), which we can be structured and exploited to provide content-based recommendations.

In this case, the weight assignment can be binary (where 1 represents the presence of a term in the document, and 0 its absence), or can be the *frequency of occurrence* of a term t_k in the document d_i and denoted as $tf_{k,i}$ (see Equation 2.4). In order to avoid a predisposition towards long documents, the measure $tf_{k,i}$ is usually normalized by dividing $f(t_k, d_i)$ by the maximum frequency of any term in the document.

$$tf_{k,i} = \frac{f(t_k, d_i)}{max_{\forall t}\{f(t, d_i)\}} \tag{2.4}$$

Another possible measure to use is the *inverse document frequency* of each term, which for the term k can be denoted as idf_k (see Equation 2.5), where $\mid M \mid$ is the number of documents existing in the collection M and $\mid n_k \mid$ is the number of documents where the k-th term appears. The logarithm is applied to smooth the resulting value of the division. The goal of this measure is to give less importance to common terms. In this case, a term t_k with low occurrence frequency in the collection of documents M is more discriminating than a term that appears in all the documents. Hence, the idf_k value will be high for rare terms (or terms with a small number of occurrences in the complete collection of documents) and low for the very frequent ones [MRS08].

$$idf_k = log\frac{\mid M \mid}{\mid n_k \mid} \tag{2.5}$$

An evolution of the above measures is the combination of both, $tf_{k,i} \times idf_k$, which establishes a relation between the term frequency $tf_{k,i}$ within a document and its frequency in the documents of the collection idf_k. In order to normalize the frequencies for long and short documents, the equation would be as shown in Equation 2.6.

$$tf_{k,i} \times idf_k = \frac{f(t_k, d_i)}{max_{\forall t,d}\{f(t, d)\}} \times log\frac{\mid M \mid}{\mid n_k \mid} \tag{2.6}$$

2.2.2.2 Advantages and Disadvantages of Content-Based Recommendations

One of the advantages of this type of recommendation approach, in comparison with collaborative filtering methods, is the user independence: unlike in content-based recommendation methods, in collaborative filtering it is necessary to know the ratings from other users for the recommendation of items that are mostly liked by the neighbors (user-based CF) or items data have similar ratings to the ones mostly liked by the user (item-based CF). This implies that a content-based recommendation system is able to suggest items that have not yet been rated by other users. Furthermore, content-based recommendation systems facilitate the explanation of why certain items have been recommended to the user, by using the descriptors of the items.

A disadvantage of content-based recommendation methods is the need to define and specify the information that characterizes the items (e.g., music, artworks, etc.). Besides, ignoring the information of other users has the disadvantage that the system will recommend only items similar to those already rated by that user, excluding

from the recommendation list the novel, diverse and unexpected items that could be of interest to the user. The latter disadvantage is commonly known as the *overspecialization problem*. In addition, the *new user problem* is present. A content-based recommendation system will not be able to suggest recommendations to a user when few ratings have been provided by that user, unless her/his profile has been explicitly defined.

2.2.3 Hybrid Recommendations

Hybrid recommendation systems are obtained through the combination of two or more recommendation techniques [Bur02, Bur05, Bur07]. Most of the time, hybrid recommendation algorithms are motivated by the need to increase the quality of the recommendations. For example, the combination of collaborative filtering and content-based recommendation methods helps to handle certain limitations that they have when used separately [MCG+99, SN99, KLP+06]. Generally, for example, the combination of collaborative filtering and another recommendation technique can help with the cold start problem [SPUP02, LVLD08].

There are different ways to combine recommendation methods into a hybrid recommendation system. For example, in [Bur02, Bur07, LST12, HLGZ14, SS14] the following classifications are proposed:

- *Weighted*: the predicted rating of an item is determined from the results of all the recommendation models applied separately in the system. For example, the P-Tango system [MCG+99] uses a weighted hybrid approach that combines content-based and collaborative filtering recommendation models to provide personalized access to daily news stories. In this system, the rating prediction is based on a weighted average of the predictions obtained from both models. In addition to the weighted average, other voting strategies may be applied, such as the sum, minimum, or maximum, of all the item ratings estimated by the recommendation models applied [HLGZ14].

- *Switching*: a criterion to switch between several recommendation models is applied. For example, the DailyLearner system [BP00] uses a content-based and collaborative filter recommendation models, where a content-based recommendation model is employed first. If this model cannot predict a recommendation with sufficient confidence, then a collaborative filtering recommendation model is applied. Similarly, the movie recommendation system MoRe [LC08] implements a hybrid approach by switching between content-based and collaborative filtering recommendation models.

- *Mixed*: recommendations obtained from different recommendation models (applied at the same time) are presented together. For example, the PTV system [SC00], that recommends TV programs, uses a mixed hybrid recommendation approach that assembles content-based and collaborative filtering recommendation models. It bases its recommendations on the programs that a user

has liked in the past, by using the textual descriptions of TV shows (content-based recommendation) and on the programs that similar users liked (collaborative filtering recommendation). The items recommended by both approaches are collected into a single list.

- *Feature combination*: features relevant to different recommendation models are considered. For example, in [BHC98] the authors formalized the recommendation problem as a learning problem. The system applies an inductive learning approach that combines features used by collaborative filtering (e.g., item ratings on a scale of one to 10) and content-based approaches (e.g., certain content information of the items). From a binary classification perspective, the input of this approach are features that include user-item ratings and item content information, and the output is a set of items that the system predicts that will be liked by the user. Therefore, the target classes are *liked* or *disliked*, rather than ratings.

- *Cascade*: the set of candidate items obtained by one recommendation model is refined by another. The first recommendation model is used to produce a coarse ranking of candidates, while the second model focuses only on those items for which additional refinement is needed. Hybrid recommendation systems in cascade are motivated by social processes, such as when someone selects several items of interest and asks for external opinions about them to others, in order to ensure the best selection. For example, in [LST12, LST14], the problem of recommendation is decomposed as a two-level cascade recommendation approach. The first level applies the content-based features of items to incorporate the individualized user preferences within the recommendation process, by using a one-class classifier. In the second level, a collaborative filtering model is used to select appropriate items resulting from the first level. Another example is the restaurant recommendation system EntreeC [Bur02], which first obtains a set of restaurants that match a user preferred cuisine (e.g., Chinese, Vegetarian, etc.) by using a knowledge-based model, and then those restaurants are ranked with a collaborative recommendation method.

- *Feature augmentation*: it is similar in some ways to the feature combination, but in this case the rating produced by one recommendation model is used as an input feature in another recommendation model. For example, the content-based book recommendation system Libra [MR00] utilizes the rating prediction obtained through collaborative filtering as the input to a content-based recommendation model. A similar example was presented in [WSS03, SSW04], where the authors used association rules over collaborative data to obtain new content features for content-based recommendation. In the same line, a hybrid collaborative recommendation model based on Bayesian networks, that combines both content and collaborative features, was presented in [CFLHRM10].

- *Meta-level*: it is inspired by an idea similar to the one behind the feature augmentation strategy. In both cases, the recommendation model learnt is used

as input to another one. However, in an augmentation hybrid recommendation system, the features generated by the model learnt are the input to the second model, while in a meta-level hybrid recommendation system the entire model becomes the input. For example, LaboUr [SKK04] is a framework that generates content-based user profiles, by using instance-based learning, which are then compared collaboratively.

The hybridization of several recommendation models can alleviate problems that appear when using pure recommendation systems. In Table 2.3, we summarize some of the benefits of hybrid recommendation approaches mentioned in [Bur02, Bur07].

Hybridization strategy	**Benefits**
Weighted	It is one of the simplest hybrid strategies to implement.
Switching	It alleviates the weaknesses of the recommendation algorithms used, highlighting their strengths. Nevertheless, the switching criteria generate an additional complexity during the recommendation process.
Mixed	It avoids the "new item" problem. A content-based approach facilitates the recommendation of items based on their features even if they have not been rated by anyone. However, it does not solve the "new user" problem, since both the content-based and collaborative filtering recommendation models need some data about the user preferences.
Feature combination	It reduces the sensitivity of the system to the number of users who have rated an item, since the system considers collaborative data without depending on it.
Cascade	It is useful in cases where an additional discrimination of the items to recommend is necessary. This strategy is more efficient than the weighted hybridization, which applies all its recommendation models to all the items.
Feature augmentation	It would be applied when there is a desire of additional knowledge sources. Generally, the augmentation can be done offline, making this strategy attractive. Moreover, the use of the feature augmentation approach is usually preferred to a feature combination, as the augmentation hybrid approach includes a smaller number of features as input to the main recommendation approach.
Meta-level	It avoids the problem of data sparsity, by compressing the user's interests into a model learnt (e.g., content-based). In this way, a second recommendation mechanism (e.g., a collaborative recommendation model) could operate on this representation more easily than on a traditional user profile.

Table 2.3: Benefits of hybrid recommendation approaches.

There are only a few studies that compare hybrid recommendation processes with experiments using the same data [Bur02]. For example, the meta-level and weighted approaches based upon content, collaborative, and demographic data, were compared in [Paz99]. In terms of precision, both hybrid recommendation approaches led to significant improvements. Moreover, a detailed analysis of approaches for the development of hybrid recommendation systems was presented in [Bur05].

2.2.4 Similarity Measures

As discussed previously, some recommendation approaches (approaches based on memory CF) use similarity measures. For example, content-based recommendation models apply similarity measures to determine the closeness between two weighted feature vectors, whereas in user-based collaborative filtering the similarity between rating vectors specified by users is measured. Among the different existing measures to determine the similarity between two objects (e.g., users or items), we can highlight the following ones:

- *Euclidean distance.* It is one of the simplest and most commonly used distance measures [JAK16]. The distance equation is as follows:

$$distance(p, q) = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2} \tag{2.7}$$

  where n is the total number of features, and p_k and q_k are the k-th features of data objects p and q, respectively. The minimum distance between two objects is 0, whereas a value very far from 0 represents that the objects are very different. The resulting value of computing $1 - distance$ can be considered a metric of similarity.

- *Jaccard coefficient.* It determines the degree of similarity between two sets of items [RV96], obtaining values between 0 and 1. The value of 1 indicates the equality between the two sets. The Jaccard coefficient is calculated as follows:

$$jaccard(I_u, I_v) = \frac{\mid I_u \cap I_v \mid}{\mid I_u \cup I_v \mid} \tag{2.8}$$

  where I_u and I_v represent the item sets preferred by the users u and v, respectively.

- *Cosine measure.* It considers each object as a feature vector and defines the similarity as the cosine of the angle between two feature vectors in the vector space [SM86] where $\bullet$ denotes the scalar product of vectors p and q:

$$cosine(p, q) = \frac{(p \bullet q)}{\|p\| \times \|q\|} \tag{2.9}$$

  If the cosine of the angle is 1, it means that the objects are very similar (equivalent in their features). Otherwise, values close to 0 will imply that the vectors are orthogonal, and so the objects are completely different (the maximum possible separation in the vector space).

- *Pearson correlation coefficient.* It determines the similarity through the linear correlation between the two objects [BCHC09], obtaining values between -1 and

+1. A positive value indicates that the values of both objects are correlated. The Pearson correlation is calculated as follows:

$$pearson(p,q) = \frac{\sum(p,q)}{\sigma_p \times \sigma_q} \tag{2.10}$$

where σ is the standard deviation of the items and $\sum(p,q)$ is the covariance of the data points p and q.

The Pearson correlation is one of the similarity measures most commonly used in collaborative filtering. Several empirical studies show its successful utility in obtaining the best neighbors [BHK98, HKR02, HF08]. In [BHK98], the authors found that the Pearson correlation performed better than the cosine similarity. Some variations of the Pearson correlations in memory-based collaborative filtering can be seen in [MH04]. Moreover, the cosine similarity measure is the most widely used in content-based recommendation systems to predict the user's interest in a particular item [RRSK11].

2.3 Next Generation of Recommendation Systems

Many studies have been performed in the area of recommendation systems [RRSK11, BOHG13]. Despite this, the research community stills continues developing new approaches, as many interesting problems remain unsolved, such as how to obtain suitable recommendations in the absence of previous information about the user, how to ensure enough variability in the items recommended, how to incorporate contextual information into the recommendation process, how to combine and exploit information from other sources such as social networks, etc. [AT05, FJN+13].

In this section, we focus exclusively on context-aware recommendation systems as the new generation of traditional recommendation systems. First in 2.3.1, we explain the main context-aware recommendation paradigms described in the literature (pre-filtering, post-filtering, and contextual modeling). Nowadays, combining CARS with mobile computing is an important challenge in this area. Hence, we also treat aspects related to both areas in Section 2.3.2. Finally, in Section 2.3.3 we discuss the basics of location-aware recommendation systems as a specific case of CARS.

2.3.1 Context-Aware Recommendation Systems

Most RS operate in a 2D $User \times Item$ space. However, considering only information about the users and items is not enough in applications such as the recommendation of vacation packages. In this case, it is important not only to determine which items should be recommended, but also when these recommendations should be provided and how to combine them in a ranked list. Moreover, traditional collaborative filtering techniques generally take into account all the collected ratings of the items to generate the recommendation models; these techniques assume that the context is homogeneous, but actually a user can assign different ratings to the same item in diverse contexts, as the relevance and interest of a specific item for a user may depend

on her/his current context. Therefore, additional contextual information (e.g., the time, with whom the user is with, the weather conditions, what the user is doing, etc.) should be considered in the recommendation process.

With advances in the fields of ubiquitous and mobile computing, the lack of analysis of contextual information in recommendation systems has been strongly criticized [AT05, WS07, AT11, MP14]. So, whereas researchers and developers had previously mainly focused on solving classic problems of recommendation systems, such as the *cold start* problem [SPUP02, Ahn08, LVLD08, ZZX13, Son16, XYT+17, WHC+17], *high dimensionality* [NIIZ15, Sym16], *spam vulnerability* [MHF07, VK16], and many others, researchers working on recommendation systems have recently recognized the need to investigate them in domains where the contextual information is particularly relevant [AMRT11, CRC16, CCZ17, RSHS17].

The integration of recommendation systems and context-aware computing (see Section 2.2 and 2.1.3, respectively) has given rise to the so-called *Context-Aware Recommendation System*. A pioneer proposal in the field of CARS is the one by Adomavicius et al. [ASST05, AT08, AMRT11]. In order to improve the recommendations based on contextual information, the authors of that work extend the classical 2D paradigm to a multidimensional recommendation model that provides recommendations based on multiple dimensions: $(User \times Item \times Context)$. So, besides considering the information of users and items, CARS take into account the context information, which is a set of contextual attributes $C = \{c_1, c_2, \ldots, c_q\}$. In particular, those authors introduced three different context-aware recommendation paradigms:

- *Pre-filtering*, where the contextual information is used to filter the data before applying traditional recommendation algorithms.
- *Post-filtering*, where the contextual information is considered only in the final step of the process. So, contextual information is initially ignored and the ratings are predicted using any conventional 2D recommendation system, taking all the potential items to recommend into account. Afterwards, the resulting set of recommendations is adjusted (contextualized) for each user by using contextual information.
- *Contextual modeling*, where the contextual information is used directly in the modeling technique as part of the estimation of ratings.

The pre-filtering and post-filtering methods consider the context as an additional filtering step that can be applied to any traditional recommendation algorithm, either to restrict its input (pre-filtering) or its output (post-filtering). On the other hand, contextual modeling recommendation systems imply a radically different approach, as the contextual information directly affects the generation of the recommendation models. Later in this section, we will discuss in more detail the three paradigms separately.

In several studies, the pre-filtering, post-filtering and contextual modeling paradigms have been compared [PTG+09, PG11, CFTCD13, PTG14a, PTG14b]. The most recent experimental analysis shows that none of the considered context-aware

recommendation paradigms dominates the others in terms of their predictive performance and diversity measures [PTG14b]. However, the performance was affected by several factors, such as the type of recommendation task (e.g., find all or only the top-k items), the granularity of the context information, and the type of dataset (e.g., depending on features such as a high or low sparsity and heterogeneity of data).

2.3.1.1 Pre-filtering Paradigm

In the *pre-filtering* approach, the contextual information is used to select the most relevant data ($User \times Item$) for 2D recommendations. In Figure 2.5, we show the general process of the pre-filtering paradigm. Firstly, the contextual information is used to filter out irrelevant ratings. Then, a traditional recommendation model based on contextualized data suggests items to the user.

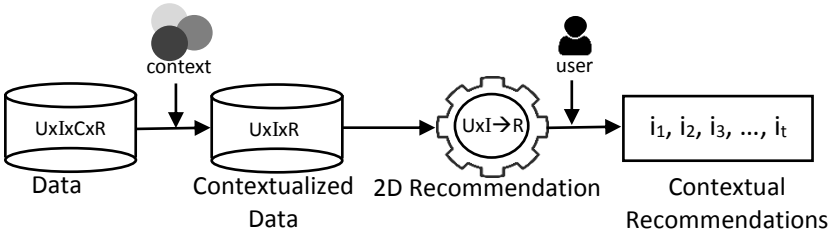


Figure 2.5: Pre-filtering paradigm.

The pre-filtering paradigm is also known as the *reduction-based approach*, as it reduces the problem of multidimensional contextual recommendations to the traditional 2D recommendation space [ASST05]. For example, in a context-aware music recommendation system, if a person enjoys listening to music while running and is at a action moment practicing sports, then the recommendation system will only use rating data ($User \times Item \rightarrow Rating$) related to the context *running*.

An advantage of this approach is that it supports any of the 2D recommendation models proposed in the literature [AT05]. However, a poor amount of filtered data could generate a severe constraint, since the model would not have enough data to generate confident recommendations. Moreover, a future work in the pre-filtering paradigm would be the incorporation of context hierarchies. For example, the context $C = \{Girlfriend, Theater, Saturday\}$ can be generalized to $C' = \{WithCompany, AnyPlace, AnyTime\}$.

2.3.1.2 Post-filtering Paradigm

The basic idea of the post-filtering approach is to consider the context as an additional constraint to verify a posteriori. As shown in Figure 2.6, this paradigm does not take into account the contextual information in the initial data input of the 2D recommendation model. Only the ranked list of candidate items (obtained by using a traditional

2D recommendation model) will be adjusted with the contextual information. This adjustment can be performed in two ways [AT11]:

- *Filtering* (or selecting) the most relevant items in a given context. In a context-aware book recommendation system, an example of item filtering would be the following: if a person usually reads science books over the weekend, the system may remove non-science books from the candidate list of books to recommend.
- *Adjusting the ranking* of the list retrieved based on a given context. Following with the same scenario, if the ranking adjustment strategy is applied instead, books with more stars (i.e., better valued) of the writers preferred by the user (in that specific context) would have a higher value in the ranked list.

In the filtering adjustment, if there are very few contexts similar to the one of current user, then many items from the candidate list to recommend could be eliminated (even all), in the worst case. In the case of ranking adjustment, if there are no similar contexts, then an approach equivalent to a traditional recommendation would be applied.

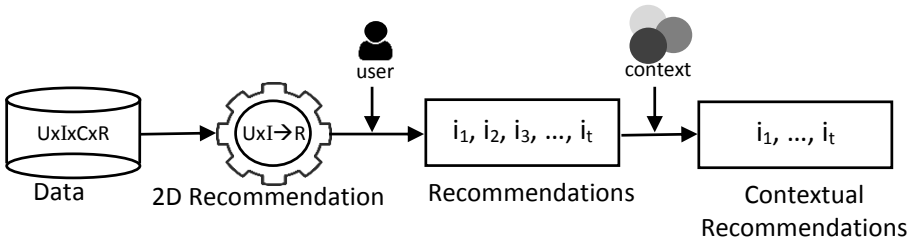


Figure 2.6: Post-filtering paradigm.

Moreover, the contextual post-filtering approaches (for both forms of adjustment) can be classified into the following types [AT11]:

- *Heuristic* post-filtering approaches, which try to find the common item features for a user in a given context, and then use these features to adjust the list of recommendations.
- *Model-based* post-filtering approaches, which build models to predict the probability with which the user prefers an item type (e.g., likelihood of choosing books of a certain literary genre) in a given context, and then use this probability to adjust the list of recommendations.

As we indicated when describing the pre-filtering paradigm, a relevant advantage of post-filtering paradigm is the ability to use any of the traditional recommendation models. In addition, similarly to the pre-filtering approaches, incorporating the ability to handle context generalization models (context hierarchies) into post-filtering paradigm is an interesting future work.

2.3.1.3 Contextual Modeling Paradigm

In the contextual modeling approach, the contextual information is used directly in the recommendation model. For this purpose multidimensional (MD), predictive models (e.g., a probabilistic model, decision tree, etc.) or heuristics that incorporate the context dimension to the user and item data are applied (see Figure 2.7). This contextual approach assumes that context attributes are appropriate features to learn a recommendation model.

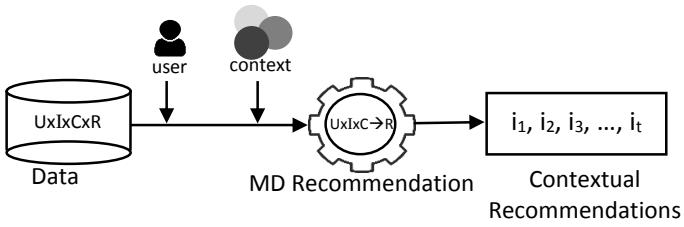


Figure 2.7: Contextual modeling paradigm.

In this paradigm, traditional 2D recommendation algorithms cannot be used directly (unlike in the case of the pre-filtering and post-filtering paradigms). However, these can be modified (or extended) with the purpose of incorporating the context dimension in the rating estimation. For example, the traditional neighborhood-based recommendation approach [BHK98] was extended to the multidimensional case in [ASST05].

2.3.1.4 Challenges of Context-Aware Recommendation Systems

Some challenges of context-aware recommendation systems (see Table 2.4) are inherited from traditional recommendation systems (e.g., sparsity, cold start, high dimensionality problems, security, privacy, and spam vulnerability), while others arise as new ones. Among the additional challenges, the following are open research issues in the literature of CARS [YL10, AMRT11, VMO+12]:

- **From the perspective of context-aware computing:**
  - The variety of application scenarios and user needs make it difficult to determine what types of contexts are actually needed in CARS. Hence, the *efficient discovery of valid (or suitable) context types* for a specific domain is a serious challenge that CARS should overcome, in order to reduce the difficulty of context acquisition and the computational cost of recommendation algorithms, thus improving the performance accuracy of CARS. According to [ASST05], this challenge can be treated as a problem of feature selection to reduce the dimensionality of the context, and thus make context comparisons more efficient.

- *Context acquisition and automatic discovery of dynamic user preferences* from several external data sources (e.g., social networks, sensors, RFID data, etc.) is a major research challenge for CARS. The resulting recommendations could be more effective if the characteristics of dynamic environment were effectively exploited.
- Another critical issue for CARS is the *development of generic contextual models*. The problem of the current proposals (e.g., [YZZ+06, SB10, SMP+11]) is that they model information for a very specific application domain (e.g., tourism, movies, etc.) or more abstract domains (e.g., products, web services, e-learning, etc.), and so their domain-specific models cannot be easily reused in other recommendation scenarios. However, in the literature of CARS there are some works that try to solve this challenge. As an example, [LD06] presents a generic model using an ontology, which can be used in different types of recommendation systems, and model data, context, and the recommendation process itself. Moreover, [MP13] carried out a study to try to determine whether a more generic modeling approach could be applied for CARS. As a result of the study, the authors proposed a novel generic contextual model for CARS, which was theoretically evaluated with positive results.
- Very few works in the CARS literature combine the context history and the user behavior [PTG08, HSKK09]. Hence, *the understanding of the user's behaviors with context history* should be improved. This could be very important to improve the performance accuracy.

- **From the perspective of mobile computing:**
  - The existing *gap between CARS and mobile computing* is a serious challenge that CARS should face. For example, mixed RS and mobile computing solutions would be useful in the case of a user who is walking down the street and uses a mobile application that suggests to her/him an appropriate taxi in real-time (in this case, both the user and the target items may be moving).
  - *User interfaces designed for recommendation purposes* (explicit or implicit recommendations) should be simple and easy to understand. However, very few studies have evaluated the usability of interfaces in the context of recommendations [BSW11, SBCH12, GWH13, BKR13, HWBH16], or have studied in depth the best way to present the information to recommend.
  - Development of *generic and flexible architectures* that facilitate the creation of context-aware recommendation systems for mobile environments. This aspect should be analyzed, given the interest of having a generic solution that can be extended and adapted to different applications and domains [dCRHI14c].
  - Most context-aware recommendation systems require users to explicitly express their interests and information needs as a query. Currently, due

to the limitation of mobile devices (e.g., data input is inconvenient), a key challenge is how to *proactively deliver relevant recommendations* to the user's mobile device [VWB11, WHBGV11, LW12, GWH13].

- **From an overall perspective:**
  - The *lack of public datasets available* and the emergence of *new evaluation measures* different from the classical ones (e.g., MAE, RMSE, precision, recall, and F-measure score) adjusted for the evaluation of CARS, such as the combination of metrics (e.g., the accuracy and the diversity with the latency) or the incorporation of context parameters in existing measures, are currently critical challenges. In Section 2.4.2, we explain in more detail both challenges.
  - *Development of practical context-aware recommendation applications*, especially in real business domains. Most research on CARS has been conceptual. Generally, researchers implement a context-aware recommendation model, which is tested (using datasets) and compared with other models proposed in the literature of the same research field. There has been little work done on developing practical applications for CARS [AT11].

In certain occasions, the problems to be solved require the combination of several recommendation techniques. Hence, the combination of context-aware recommendation paradigms facilitates the emergence of new approaches. An example is the approach proposed in [ASST05], which uses several pre-filtering models and combines their outputs. Another interesting hybridization could be to combine the pre-filtering and post-filtering paradigms. For example, sometimes it may be more useful to use the pre-filtering approach for attributes such as the day of the week, while for context attributes like the weather the post-filtering approach might be more appropriate.

2.3.2 Context Aware Recommendation Systems in Mobile Environments

In the last years, some context-aware recommendation architectures have been proposed in the literature [BHHD13, HLGZ14, ZMB15]. However, these architectures are not designed with mobile users in mind, where the context and the movements of the users may be important factors to consider when deciding which items should be recommended. This problem is indeed currently a future research direction.

Moreover, the widespread availability of mobile devices, such as smartphones and portable computers, implies that the relevance of mobile computing scenarios is nowadays undeniable. This, in turn, demands new approaches for the development of recommendation systems that can handle and effectively exploit the data available in those environments. Hence, the combination of *context-aware recommendations* and *mobile computing* gives rise to the emergence of *Context-Aware Mobile Recommendation Systems* [WBE09, VMO+12, LMCX13].

Challenges	State of the art
1) **Automatic data acquisition and context exploitation**: representation, acquisition, and enrichment of data dynamically.	CARS could be more effective if the characteristics of the dynamic environment were effectively captured and exploited. Examples of related contributions: -Exploiting GPS trajectories: [CSdVR11, TS05, TS06a] -GPS sensing: [BZM12, BKR13]
2) **Evaluation**: evaluation measures adjusted to dynamic environments, context-enriched data sets.	There is a need to use evaluation measures different from the classical ones, adjusted for the evaluation of CARS. Moreover, the datasets used for evaluation are usually still the same datasets used to evaluate traditional recommendation systems (e.g., MovieLens and Foursquare). Examples of related contributions: -Diversity measure: [SU11] -Usability questionnaire: [BKR13, KCL09, SBCH12] -Continuous query processing performance: [LSEM12]
3) **User interfaces**: proper design of user interfaces for mobile devices and dynamic environments.	It is necessary to design suitable user interfaces (i.e., simple and intuitive) for CARS, in order to avoid overloading the user with information. Examples of related contributions: -Usability evaluation of interfaces: [BKR13, SBCH12]
4) **Security and privacy**: ensuring the location privacy and user security.	The study and application of techniques to ensure location privacy and user security are important challenges to consider in the development of CARS. Examples of related contributions: -For recommendation systems in general: [RKM$^+$01]
5) **Generic architectures and middleware**: emergence of generic architectures.	Despite the efforts, there is still no implemented architecture that facilitates the development of CARS for mobile environments. An adaptable architecture that could be extended and customized for several application scenarios would be really useful. Examples of related contributions: -Proposal of a generic framework: [dCRHI14c].

Table 2.4: Summary of key challenges related with CARS.

The main goal of context-aware mobile recommendation systems is to suggest the right items (or services) to mobile users at anywhere and anytime, being the

contextual information a key element in determining the relevance of the items. From the perspective of mobile computing, these recommendation systems are characterized by the following elements [LMCX13]:

1. *User mobility*: the users can access a mobile information system in different locations, while moving.
2. *Device portability*: the device used to access the information system is mobile (e.g., a smartphone, a portable computer, etc.).
3. *Wireless connectivity*: the device used to access the recommendation system uses wireless communication technologies (e.g., Wi-Fi or Bluetooth).

In mobile scenarios, the context is highly changing. For example, in a taxi recommendation scenario both the users and the items to recommend can be on the move. Thus, mobile recommendation systems must automatically update the contextual information of users and items. Likewise, during the recommendation process, the answer to a user's query must be continuously re-evaluated by the system until the user decides to cancel it. Some context-aware mobile recommendation systems have already been developed, but they focus mostly on specific domains (e.g., restaurants, museums, gas stations, supermarkets, foods, etc.) [ONMU06, LWGD07, HPNM09, WHBGV11, VHR12]. In Section 7.2, within the context of related work, we present more examples of CARS for mobile environments.

2.3.3 Location-Aware Recommendation Systems

In context-aware recommendations, the context variable *location*, of the users and/or the items, has been proved to be of special importance to suggest relevant recommendations [HNV06]. Hence, the emergence of *Location-Aware Recommendation Systems* (*LARS*), as a special case of CARS, must be highlighted.

LARS take into account the spatial properties (locations) of users and/or items to suggest proper recommendations. The emergence of LARS comes from the fact that users typically prefer nearby items (e.g., restaurants, museums, cinemas), as the effort needed to reach items close to their physical positions will be smaller. Moreover, it may happen that only nearby items are relevant or that items located far have a short spatio-temporal relevance. For example, a suggestion about a specific parking space provided to a driver searching for parking could become obsolete in a short time if the parking space is not nearby (while the user drives towards the parking spot, it can be occupied by another vehicle).

In general, LARS can be considered as an extension of traditional recommendation systems, and an important subset of CARS, that focuses on the *location* dimension. In LARS, the rating is modeled as a function of the item, user, and location, that is, $f : U \times I \times L \rightarrow R$. The framework proposed in [LSEM12] classifies location-based ratings in three categories:

- *Spatial ratings for non-spatial items*, represented by the tuple $\langle user, ulocation, rating, item \rangle$, where *ulocation* is the user's location.

- *Non-spatial ratings for spatial items*, stated by the tuple ⟨*user*, *rating*, *item*, *ilocation*⟩, where *ilocation* represents an item's location.
- *Spatial ratings for spatial items*, represented by the tuple ⟨*user*, *ulocation*, *rating*, *item*, *ilocation*⟩. In this case, the location of the user and the location of the item are both relevant.

So, the *location* can be associated to the physical position of the user when she/he rates an item (e.g., a book rated by a user who is at home), to the location of an item (e.g., the geographic location of a restaurant rated), or to both. The set of recommendations provided to the user should be monitored and kept up-to-date, as the relevant recommendations may change due to movements of the user and/or the target items (e.g., if the items are taxi cabs).

As shown in Figure 2.8, in LARS the location information of the users and/or items can be captured in different ways (Section 2.1.2 provides more details about existing mechanisms for obtaining location information). The location information captured is employed to recommend relevant items in several application domains, such as for the recommendation of news [SBE+12, AW14], shopping [TS06a, YCD08, ZOON12], POIs [HNV06, TR09, THK11, BZM12, CFP+12, SBCH12, YCS+14], music [BKR13], and tourism [YpC09, LMZW10, CSdVR11, NBSM12]. Currently, several real-world recommendation systems use the location as an important parameter for the suggestion of relevant items. Well-known examples are Google Now [Goo12a], Foursquare [CS09], and Yelp [Yel04].

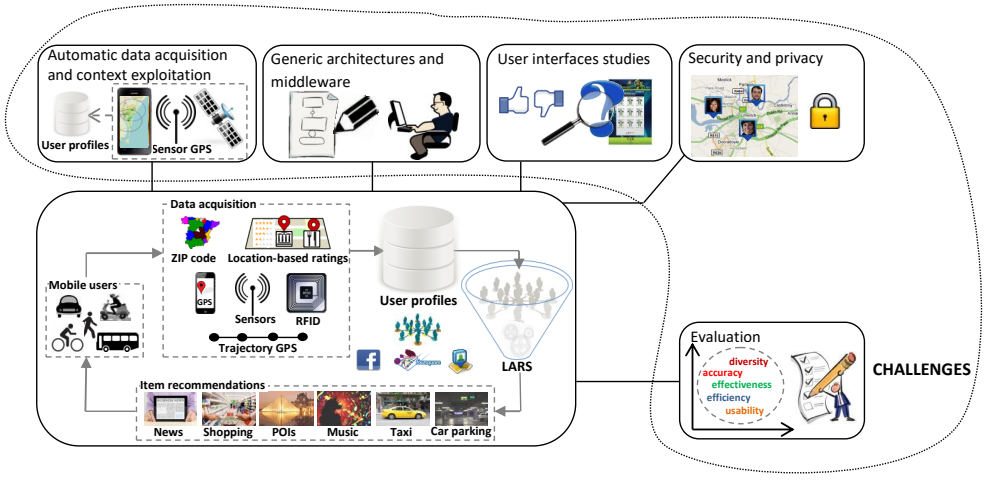


Figure 2.8: Overview of LARS.

In [dCRHITLH15], we presented a survey that explains in more detail the aspects mentioned in this section. Another review about the main components of a mobile location-based recommendation system was presented in [Lat15]. Besides, in Section 7.3, we provide more information related work on LARS.

2.4 Evaluation of Recommendation Systems

The evaluation of recommendation systems aims at verifying if a proposed system meets the proposed goal and solves the problems identified. It usually implies comparing different algorithms or recommendation systems. In general, the evaluation of recommendation systems allows measuring a variety of properties, such as the accuracy, user satisfaction, robustness, scalability, etc. [SG11, BGLB16]. The existing evaluation methodologies can be classified into two types [HKTR04, SLG11]:

- *offline*, based on a pre-collected dataset that allows simulating the behavior of users interacting with the recommendation system.
- *online*, based on experiments with real users that interact with the recommendation system in real time.

The offline evaluation, unlike the online evaluation, has the advantage that experiments are less costly and can be reproduced under the same conditions. In addition, offline experiments facilitate the comparison of several recommendation systems at different moments in time.

Generally, offline experiments are highly used because they do not require real users and allow comparing a wide range of recommendation algorithms at a low cost. This type of evaluation requires data collections for specific domains. A *data collection* is a dataset, usually tabulated, where each row represents an element (or instance) and each column is related to a feature (or attribute) that describes the element. In the field of recommendation systems, there are datasets commonly used, such as MovieLens [Gro16, HK15], the Netflix Prize dataset [Net09, BL07], the Jester dataset [Jes03, GRGP01], Yahoo! datasets (music, movie, or images) [Yah04, MZ09, SBM12], the Book Crossing dataset [Zie04, ZMKL05], etc.

Creating a dataset requires a lot of effort by a wide community of users or experts, hence the importance of publishing the existing data collections. Facilitating the use of such data collections will allow the scientific community to make comparisons between different algorithms on the same data collection, since different algorithms may behave better or worse in different collections of data.

For the evaluation of the accuracy of recommendation systems, "as for the evaluation of any machine learning algorithm" [KZP07], the data collection is commonly partitioned into two datasets: training and testing. The training set is used to learn a recommendation model, while the testing set is used to evaluate the ability to predict new ratings by using the model learnt. In this way, the accuracy of the recommendation system can be measured by comparing the predicted data with the known data. Sometimes, a third set of data (the validating dataset) is considered to adjust parameters of the machine learning model the testing set.

In [SLG11], several ways of partitioning the data collection (in training and testing) are mentioned. For example, *cross-validation* techniques are one of the most popular. In occasions, they are classified into the following three types [Koh95, AC10]:

1. *K-fold cross-validation* [BG04]. The dataset is divided into K disjunct subsets of equal size. One of the subsets is used as testing data and the rest $(K-1)$

as training data. The cross validation process is repeated during k iterations (or folds). Finally, the average of the results of each iteration is calculated to obtain a unique value. Usually, the value of $k = 10$ folds is used [MDA05].

2. *K-fold random cross-validation* [RKGP17] *or random subsampling* [DGB09]. It is similar to the first technique, but in this case, for each fold, the dataset is partitioned randomly into a set for training and a set for testing. For example, in each iteration (or fold) 30% of the data is randomly selected from the dataset for testing and 70% for training. The final result corresponds to the average of the values obtained for the different divisions.

3. *Leave-one-out cross-validation* [CT03, CT04]. It is a k-fold technique with $K = 1$. In each iteration one data instance is left out for testing, while the training is carried out on the rest of the dataset.

The first cross-validation technique is very precise, but slow from a computational perspective. Moreover, in the second technique some data may not be evaluated, and others may be evaluated more than once. Finally, the third technique has a very low evaluation error, but the computational cost is high, due to the high number of iterations needed to test the whole dataset. Hence, it is typically used only for the analysis of very small datasets.

2.4.1 Evaluation Metrics

The most commonly used metrics to determine the accuracy of recommendation models are the *Mean Average Error* (MAE) [SM95] and the *Root Mean Squared Error* (RMSE) [HKTR04, WM05], which are defined as follows:

$$MAE = \frac{1}{\mid T \mid} \sum_{r(u,i) \in T} \mid \hat{r}(u,i) - r(u,i) \mid \tag{2.11}$$

$$RMSE = \sqrt{\frac{1}{\mid T \mid} \sum_{r(u,i) \in T} (\hat{r}(u,i) - r(u,i))^2} \tag{2.12}$$

where $r(u,i)$ is a known rating of the testing set T, and $\hat{r}(u,i)$ is a predicted rating obtained by the recommendation model. Both MAE and RMSE evaluate the average model prediction error in the range from 0 to ∞. However, the RMSE gives a higher weight to larger absolute values than to errors with smaller absolute values [CD14]. This is because in the RMSE the errors are squared before they are averaged. Hence, the RMSE is more suitable when large errors are particularly undesirable.

Another popular metric is the *Hamming* distance that counts the number of ratings in which predicted and known (in the test set T) rating values disagree, as defined in the following equation [AAAPS09, KvWG09, ANPS11]:

$$Hamming = \frac{1}{\mid T \mid} \sum_{r(u,i) \in T} d(\hat{r}(u,i), r(u,i)) \tag{2.13}$$

where $d(\hat{r}(u,i), r(u,i)) = 1$ if $\hat{r}(u,i) \neq r(u,i)$, and 0 otherwise.

Sometimes, researchers need to evaluate whether the system adequately predicts that the user will use or like the recommended items, rather than if the system correctly predicts the item rating. In this case, the metrics of *precision* and *recall* can be used [GS09, Pow11].

The *precision* (see Equation 2.14) reflects the capability to recommend only useful items (avoiding the irrelevant ones), while the *recall* (see Equation 2.15) represents the *coverage* of useful items that the recommendation model can obtain (avoiding missing relevant items). Finally, as a combined metric, the *F-measure* is defined as the harmonic mean of the precision and recall [HR05, Pow11]. The *F1-measure* is a particular case of the F-measure (see Equation 2.16) for a value of $\beta = 1$. Notice that the meaning of the values TP, FP, and FN, used in the formulas of precision and recall, is explained below, in relation to the concept of confusion matrix (see Table 2.5).

$$precision = \frac{TP}{TP + FP} \tag{2.14}$$

$$recall = \frac{TP}{TP + FN} \tag{2.15}$$

$$F_\beta = \left(1 + \beta^2\right) * \frac{precision * recall}{(\beta^2 * precision + recall)} \tag{2.16}$$

To illustrate the performance of the system in terms of precision and recall, it is useful to use a *confusion matrix* (or *contingency table*), as shown in Table 2.5. This table represents the four possibilities that can arise regarding a recommendation decision. The diagonal numbers *TP* (*True Positives*) and *TN* (*True Negatives*) count the correct decisions: recommend an item when it should be recommended and do not recommend an item when it should not be recommended, respectively. On the other hand, *FP* (*False Positives*) and *FN* (*False Negatives*), represent incorrect decisions: recommend an item when it should not be recommended and do not recommend an item when it should be recommended, respectively.

		Predicted	
		Positive (relevant item)	Negative (non-relevant item)
Real	Positive (relevant item)	TP	FN
	Negative (non-relevant item)	FP	TN

Table 2.5: Template of a confusion matrix.

In order to take into account the positions of the items in the recommendation ranking, the following precision metric at k items can be used [TS06b, ML10]:

$$P@k(u) = \frac{1}{k}\sum_{k=1}^{k} rel(R_i, u) \tag{2.17}$$

where $rel(R_i, u)$ is 1 if the item i in the ranking is relevant for the user u, and 0 otherwise. This measure computes the precision of the top-k ratings.

On the other hand, for the comparison of rankings (e.g., to compare the ranking obtained with an ideal ranking), the measure of *Kendall's tau* [Ken38, FKS03] can be used, which we show below:

$$K^{(p)}(l_1, l_2) = \sum_{i,j \in P(l_1,l_2)} \bar{K}_{i,j}^{(p)}(l_1, l_2) \tag{2.18}$$

where l_1 and l_2 are the lists of recommended items to compare and $P(l_1, l_2)$ is the union set of the elements in both lists. If $\bar{K}_{i,j}^{(p)}(l_1, l_2) = 1$, it means that one of the following conditions have been met:

- The elements i and j appear in both lists but in reverse order.
- Both elements (i and j) appear in one list, but only one (i or j) appears in the other list.
- Only the element i (or j) appears in one list.

Otherwise, $\bar{K}_{i,j}^{(p)}(l_1, l_2) = 0$.

There are other metrics that could be used to evaluate recommendation systems, such as the Discounted Cumulative Gain (DCG) [JK02, Voo01, QLXL10, IKPC17]. In Table 2.6, we present a summary of the evaluation metrics discussed in this section. The last column indicates the goal of the metric.

2.4.2 Evaluation Challenges

Regarding the evaluation of recommendation systems, there are still significant research challenges to be addressed. Firstly, RS have become more complex over time, by considering new parameters during the recommendation process, such as the contextual information. The evaluation of CARS is a challenge, due to the *scarce availability of public datasets* that incorporate context information related to the ratings provided by the users. Although, there exist some datasets with contextual information to test algorithms for CARS (e.g., STS [BERS13, EBRT13]), they are usually incomplete, since most users prefer not to disclose their context when they rate items, or they may just be lazy to do so. Moreover, these datasets do not cover all the possible potential scenarios that researchers would need to prove the validity and generality of their recommendation approaches. Besides, traditional well-known datasets (e.g., MovieLens, NetFlix, or Foursquare, among others) are not suitable for the evaluation of CARS, and the obtention of new data collections is very costly. Real datasets could be collected more easily by a mobile recommendation system if the user's context data

Evaluation metric	Brief description	Purpose
MAE [SM95, HKTR04, WM05]	It measures the average absolute deviation between a predicted rating and the user's true rating.	Rating accuracy
RMSE [HKTR04, WM05]	It determines the root-mean-square error and is biased to weigh large errors disproportionately.	Rating accuracy
Hamming [AAAPS09, KvWG09]	It counts the number of ratings in which predicted and known rating values disagree.	Diversity
Precision [GS09, Pow11]	It reflects the capability to recommend only useful items.	Classification accuracy
Recall [GS09, Pow11]	It represents the coverage of useful items that the recommendation model can obtain.	Classification accuracy
F-measure [HR05, Pow11]	It measures the harmonic mean of the precision and recall.	Classification accuracy
P@k [TS06b, ML10]	It obtains the precision of a result set considering the first k items in the ranked list.	Ranking accuracy
Kendall's tau [Ken38, FKS03]	It compares the ranking of items to recommend obtained with an ideal ranking.	Ranking accuracy
DCG [JK02]	It evaluates the capacity of the recommendation system to show the relevant items in the first positions.	Ranking accuracy

Table 2.6: Summary of evaluation metrics.

were automatically detected. Furthermore, the definition of realistic synthetic data generators, or even crowdsourcing data collection through videogames (gamification), could be explored.

Secondly, researchers continue using traditional measures (e.g., MAE, $RMSE$, $Hamming$, *precision*, *recall*, and $F1$ score) to evaluate context-aware recommendation systems [HKTR04, YL10]. We believe that an interesting research direction could be the proposal of new evaluation measures, as indicated in [MRK06, dOG08]. For example, the inclusion and evaluation of *diversity* is as an important element in context-aware recommendation systems. This idea was considered for the first time in [GPT11]: the goal is that the users should be provided with recommendations that are diverse enough rather than very similar to each other, which is an idea that had been exploited before in Information Retrieval contexts [AGHI09]. Another example is the combining metrics such as the *accuracy* and the *diversity* with the *latency*, or including context parameters in existing measures, could be an interesting area to analyze. Moreover, most works focus on the evaluation of the effectiveness of the recommendations, but in mobile environments the usability and efficiency are also relevant aspects to evaluate, for example, timely suggestions could be more important than perfect suggestions but with a long delay.

Thirdly, in [AMRT11] the authors highlighted that "an important research direction is the comparison of the three CARS paradigms to detect strengths and weaknesses of each paradigm and to determine which one is better than the others and in which circumstances". In recent studies, this challenge has been partially addressed. For example, in [PTG14b] the authors performed an evaluation and comparison of the effectiveness of the existing paradigms (pre-filtering, post-filtering, and contextual modeling), considering the accuracy and the diversity of the recommendations. Others comparisons among the paradigms had been presented before in [PG11, CFTCD13].

Finally, potential temporal changes in user preferences [CCYX09, GŽB+14] due to the emergence of new products or services in the recommendation systems or changes in the user's habits, bring challenges for the evaluation of recommendation systems [YL10].

2.5 Summary of the Chapter

Along this chapter, we presented the technological context related to this thesis, including the field of mobile computing, the main features of traditional recommendation systems, context-aware recommendation systems, and location-aware recommendations systems. Besides, we also discussed the problem of evaluation of recommendation systems.

Firstly, we described the context of mobile computing, emphasizing the current increase of mobile devices worldwide. We focused on concepts and technologies such as P2P networks, sensors, and context-aware computing.

Secondly, we described the basics of traditional recommendations systems. Mainly, we presented a detailed explanation of the classic approaches of recommendation: collaborative filtering, content-based filtering, and hybrid recommendations. Regarding collaborative filtering, we emphasized two types of filtering: memory-based and model-based. In terms of scalability and precision, we highlighted the popularity of matrix factorization techniques, and particularly SVD. We also focused on similarity metrics, as they are a key element during the recommendation process. Besides, we mentioned the main advantages and limitations of traditional recommendation systems. Finally, we explained the motivation of hybrid recommendation systems and some ways of combining several recommendation models.

Thirdly, we presented the context-aware recommendation paradigm as the next generation of traditional recommendation systems. We described the main approaches for context-aware recommendation: pre-filtering, post-filtering, and contextual modeling. Besides, we emphasized the importance of taking into account the features of mobile computing scenarios, where the user's context is highly dynamic. As a special case of context-aware recommendation systems, we explained the fundamentals of location-based recommendation systems, where the contextual information is based mainly on the user's location.

Finally, we concluded the chapter discussing the main aspects to take into account during the evaluation of recommendation systems. Initially, we described the most popular evaluation metrics and their benefits. Afterwards, we presented a set of

evaluation challenges identified in the literature of context-aware recommendation systems.

Chapter 3

Context-Aware Mobile Recommendation Architecture

In the field of context-aware recommendation systems, only static contextual information is usually considered. However, the dynamic contextual information would very helpful in mobile computing scenarios. Despite this interest, the design and implementation of flexible and generic frameworks to support an easy development of context-aware mobile recommendation systems have been relatively unexplored.

This chapter describes our proposed architecture, called MOONRISE (MObile cONtext-aware RecommendatIon SystEm), specifically designed to exploit context-aware recommendations in mobile computing environments. In Section 3.1, we present a motivating scenario where a context-aware recommendation architecture like the one proposed in this thesis would be useful. In Section 3.2, we describe the design of the proposed context-aware mobile recommendation architecture. Specifically, we present the different layers of the architecture structured by levels, detailing the corresponding modules per layer.

3.1 Motivating Scenario

This section describes a sample scenario to illustrate the interest of a context-aware mobile recommendation architecture. The scenario shows some of the main benefits provided by the architecture that we propose in this thesis, as well as the interest of having a flexible and global framework for CARS aimed at mobile users. We follow a story-like style to describe the motivating scenario, which consists of two parts: shopping (see Figure 3.1), and leisure after shopping (see Figure 3.2).

Throughout the description of the scenario, we assume the existence of an application *MOcCARSin* (*MObile Context-Aware Recommendation System*), which can

be installed in mid-range smartphones. This application recommends (proactively or through explicit queries) products of interest to the user, considering an enriched user profile built based on opinions issued in the past about products in different contexts as well as exploiting additional contextual information obtained from the environment (via different types of sensors).

The example provided in the rest of this section illustrates the benefits of a context-aware mobile recommendation framework to provide useful recommendations in a mobile computing environment. Several mobile computing applications for different scenarios could be developed based on the generic framework proposed. This chapter focuses on the design of an architecture that could support graphic applications, such as the examples shown in Section 3.1.1 and 3.1.2. Although the fictitious application MOcCARSin named in the example does not exist, it shows the ultimate goals pursued in our research. Besides, in Chapter 6, we will focus on a different application scenario (the visit to a museum) and analyze the interest of mobile recommendations in that specific context.

3.1.1 Example Scenario: Shopping

Alice gets up at 7:30 am. It is Saturday and it is raining. She needs to buy some food for the week, but she does not know exactly what she should buy and where she could go. So, Alice decides to use MOcCARSin in order to receive recommendations of supermarkets where she can go, given her current circumstances. First, she introduces the keyword "food" as the query. Automatically, MOcCARSin enriches this input with additional contextual information that is relevant in her context, such as the *hour* and *day of the week*, the *weather conditions*, her *current location*, and even information about the persons that are with Alice at the moment. MOcCARSin infers that Alice is probably interested in buying food, as her potential desire to have lunch or dinner at that time of the day is considered unlikely; taking breakfast outside does not seem a plausible option either because Alice's coffee machine broadcast a "coffee ready" message a few minutes ago (by using *Machine to Machine –M2M–* communications [AHD14, CL14]). So, MOcCARSin asks Alice "Do you need to buy food?" and she confirms.

Then, MOcCARSin evaluates the different context parameters and Alice's preferences to suggest her a place where she can buy food. It knows that Alice prefers medium-price supermarkets rather than more expensive high-quality malls, small stores, or gourmet food stores specialized in specific types of products. However, in this case the closest supermarket is quite far, so going by foot there is not a good option given that it is raining. Moreover, Alice does not have a car, the public transportation system is not working yet at this time, and Alice's brother (who could drive her to the supermarket) is not with her at the moment. So, MOcCARSin considers interesting to include in the list of suggestions a small food store located a few blocks from her home. Alice finally selects a supermarket, and so MOcCARSin suggests her to take a taxi or wait until 9:00 when the bus service starts. Alice decides to take a taxi, and therefore MOcCARSin recommends her several taxi companies (ranked

Figure 3.1: Example scenario for mobile CARS: shopping.

based on ratings from other users), continuously showing on a map the up-to-date locations of taxis nearby along with contact information of the corresponding taxi companies.

Once in the supermarket, MOcCARSin automatically suggests products that she usually buys, products whose stock in her fridge is decreasing (this information was provided by her fridge before leaving home), as well as other products that she has never purchased but she might like. For each product suggested, MOcCARSin provides optional information such as its name, price, and appropriate routes to reach the supermarket's section where the product is located. If Alice selects a set of products at the same time, then MOcCARSin is able to suggest Alice a route that optimizes the retrieval of all those products. As the supermarket may include several floors, MOcCARSin exploits a hierarchical indoor map to obtain the appropriate walking directions, including (if necessary) taking the stairs or an elevator.

Information about products is obtained from a web service provided by the supermarket, as well as from other customers that may disseminate and share dynamically

(through a mobile P2P network) information about specific offers that they have seen. From the P2P perspective, each customer has locally stored on her/his mobile device information about their purchased products as well as their preferences. Customers located nearby can exchange information stored on their mobile devices, by using short-range ad hoc communications. Moreover, Alice has accepted to share information about her current shopping cart with the supermarket, which the marketing department exploits to offer her customized offers and related products (e.g., if Alice has bought strawberries, she can be offered whipped cream at a special price).

By using NFC (Near Field Communication) [COO13], MOcCARSin is able to provide Alice with information about products that she is currently observing, as the supermarket has deployed an appropriate mechanism to query this information, such as RFID [Rob06, Fin10] tags. For example, MOcCARSin can identify product features including its ingredients or caloric content, its price, and its expiry date. Alice is in a gluten-free diet, and so MOcCARSin warns her if she tries to pick up products that contain gluten. It also prevents Alice from choosing specific products whose expiry date is too close in time, taking into account her consumer habits (e.g., Alice might be used to leave products in the fridge for a long time).

When Alice considers that the shopping is finished, MOcCARSin alerts Alice that she is just three euros below the threshold that would entitle her to a free delivery at home. Therefore, as she does not have a car, she decides to buy also a couple of chewing gum packets and ask for home delivery during the afternoon.

3.1.2 Example Scenario: Leisure After Shopping

After leaving the supermarket, the weather has improved considerably. Alice starts walking home and she receives a recommendation to have lunch in a restaurant located nearby. It is a good time to eat and she is hungry. Besides, the restaurant offers Chinese food, which is Alice's preferred meal. So, she decides to accept the recommendation. MOcCARSin suggests Alice to ask Bob, Alice's best friend, to join her, as he also loves Chinese food. Alice contacts Bob and he accepts, but he is not nearby, so he takes his own car to go there. Bob types "parking car" in his MOcCARSin application along with the address of the restaurant, to try to find available parking spaces near the restaurant. MOcCARSin collects and offers him information about available parking spaces that may be relevant (obtained from repositories with information about parking lots and garages, data collected from parking sensors, etc.). Some of the private parking spots suggested support booking and electronic payment, which Bob finds convenient.

In the meanwhile, Alice decides to buy a present for Bob. He loves books, so she asks for appropriate book recommendations. As Bob is sharing with Alice information about his preferences and the books that he has already read, the MOcCARSin application in Alice's device uses those data to suggest Alice an appropriate book for Bob.

At the restaurant, Bob invites Alice to go to the cinema after lunch. Alice accepts, but she has the responsibility to choose the movie. By simply introducing the

Figure 3.2: Example scenario for mobile CARS: leisure after shopping.

keywords "cinema movies", MOcCARSin is able to provide recommendations about interesting movies being shown in cinemas in the vicinity. User preferences related to the price, type of cinema (preferred size, audio and video capabilities offered, comfort and additional services, etc.), and movie genres, are automatically analyzed during the recommendation process. For example, MOcCARSin excludes 3D movies, as they usually give Alice a headache. Moreover, as Alice shares many interests with her friends, she trusts particularly the opinions (ratings) provided by people in her social network, which is also taken into account by MOcCARSin when assessing the reliability and relevance of the available ratings.

3.2 Architecture

MOONRISE is an architecture designed to facilitate the creation of context-aware recommendation systems for mobile computing environments [dCRHI14c, dCRHI16]. In Figure 3.3, we show a high-level view of the architecture. It provides several traditional recommendation algorithms (see Section 2.2 for a description of their fundamentals), such as *collaborative filtering* (e.g., user/item-based recommendation [BHK98] and *SVD* [Bra03, SKKR02]), and *content-based recommendation* algorithms [LGS11a]. *Knowledge-based recommendation* algorithms [Bur07] also fit in the proposed architecture; they could be used to recommend items affected only by the user's needs and preferences but not by ratings (e.g., in a scenario where parking spaces are recommended to a driver, the suitable recommendations are usually based on objective criteria rather than ratings). Besides, it is possible to combine several recommendation algorithms in order to improve the accuracy of the recommendations; for hybridization of recommendations, several strategies can be applied [Bur07], such as *weighting* (applying several algorithms at the same time and weighing their scores to obtain an overall score) and *switching* (choosing the most appropriate recommendation model based on the available information). In addition, the architecture accommodates different context-aware recommendation paradigms (*pre-filtering*, *post-filtering*, and *contextual modeling*) in a privacy-preserving way, by sharing data only in a local area and keeping sensitive data and personal preferences privately on the user's device.

The architecture is suitable for both *indoor* and *outdoor* scenarios, as the specifics

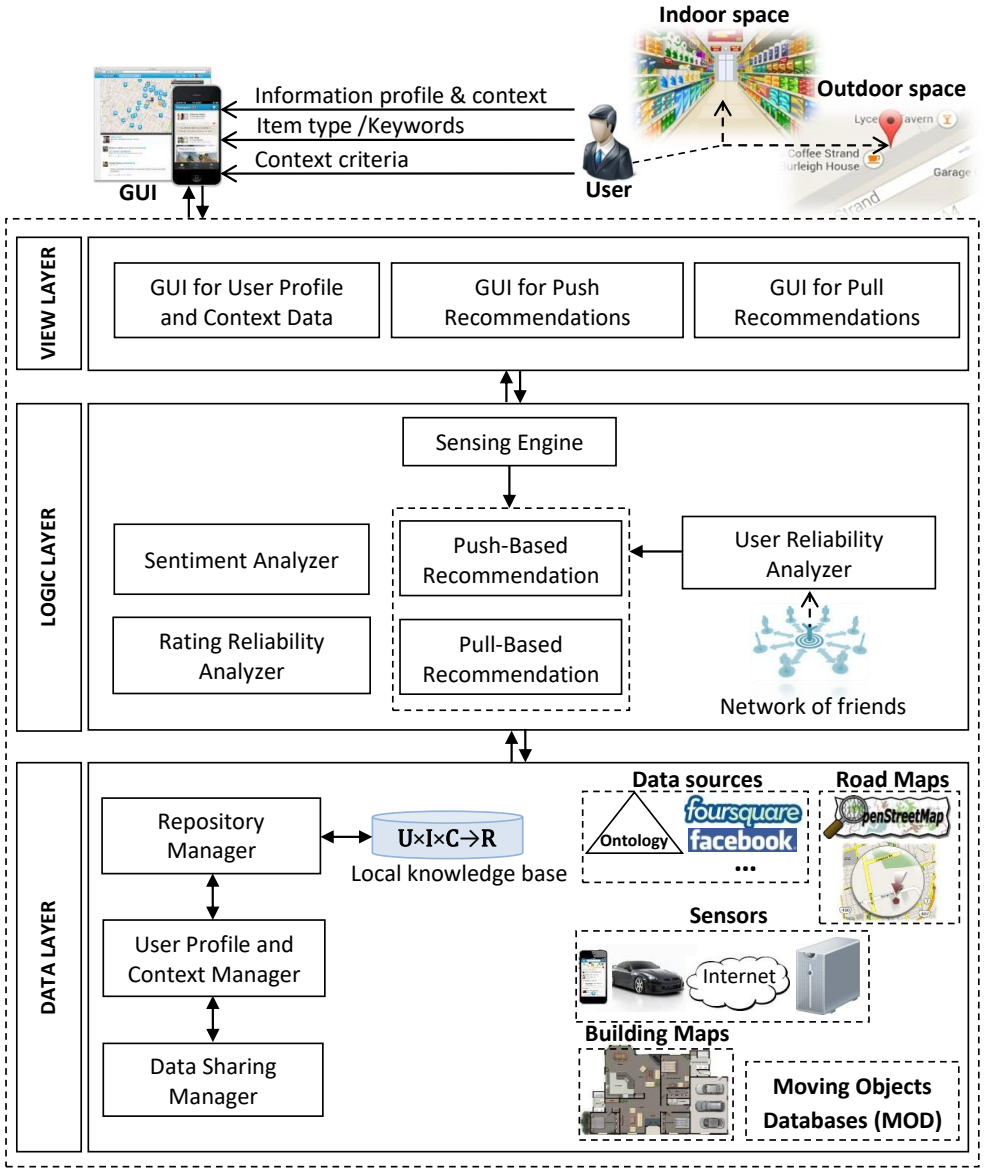


Figure 3.3: Context-Aware Mobile Recommendation Architecture.

of each environment would be exploited internally by the system (e.g., by considering appropriate positioning technologies to obtain the location of the user in that environment, to determine if the mobility of the user is completely free or constrained to specific path networks such as roads or a building's layout, etc.). Besides, no

assumption is made regarding the availability of complete information in the local knowledge base of the user's device, the communication technology used (Wi-Fi, 3G, etc.), or the possibility to query central servers and/or opportunistically benefit from information available in other mobile devices through a mobile peer-to-peer architecture [LW08]. The architecture should adapt to the available options. Whatever the scenario, the user should receive appropriate recommendations for her/his current context in real-time, by following a best-effort approach.

The *sensing engine* module implicitly exploits information from sensors and other available data sources (e.g., geospatial information services, social networks, web services, etc.) to obtain relevant context information. Dynamic context values, such as the transport way, mobility (e.g., sensed by accelerometers), temperature (e.g. sensed by temperature sensors), and time of the day (e.g., sensed by optical sensors), as well as the location (e.g., sensed by a GPS receiver) of the current user or item may be captured from sensors of mobile devices. Thus, today's mobile devices (e.g., smartphones) come equipped with a large number of sensors, which would be useful to obtain contextual information of the user's or item's environment (e.g., timer, humidity sensor, microphone, compass, gyroscope, camera, etc.) [LML$^+$10, CC12]. Moreover, a supporting infrastructure of sensors may be in place (e.g., to obtain temperature information from sensors deployed in a specified area). In [IHTLdCRH15], we have performed an in-depth study of the role of sensors in mobile context-aware recommendation systems. In our architecture, the static context information (e.g., user preferences about item features, and basic personal data such as the age, genre, etc.) obtained from the user, as well as dynamic context data (e.g., transport way, location, etc.) obtained from the environment, are considered in the context-aware recommendation process. The architecture is very generic in terms of the context model used, as we consider the generic context model proposed in [MP13], which can be adjusted to any domain.

Both *push-based recommendations* (proactive recommendations, received without explicit requests from the user) and *pull-based recommendations* (reactive recommendations, obtained as an answer to a query submitted by the user and evaluated by the system as a continuous query) are supported. In both cases, the system could simply query data available on the local knowledge base (data pushed from other mobile peers, introduced by the user, or obtained from other services and cached locally) and/or access other data repositories, as required.

The proposed architecture includes strategies to deal with well-known problems in recommendations systems, such as: the problem of *cold start* and *sparse data*, by supporting hybrid recommendations models (e.g., combining content-based and collaborative filtering) and the possibility to exploit data available outside the local knowledge base (obtained through a mobile peer-to-peer architecture and/or by directly contacting central servers or repositories); the problem of *high dimensionality*, by employing the SVD model; the problem of *vulnerability spam*, by allowing the use of techniques to identify false information injection attacks [WMB07, MBBW07, MHF07, ZLWL09, SKY11, VK16] as well as to discard information judged as unreliable through a *reliability analysis* (e.g., collaborative filtering can be constrained to

consider only, or assign more confidence to, information provided by friends [AMT05]); and the problem of *too similar or redundant recommendations*, by ensuring diversity through the use of similarity measures and clustering techniques. Moreover, it considers the possibility to obtain numerical ratings for items from opinions expressed in natural language, by exploiting supervised learning methods for *sentiment analysis* [LClC06, Liu10, Liu11, LZ12, BKKK16, YSZ17].

3.2.1 View Layer

This layer reflects the main components of the user interface. Through this interface, the user can perform the following main actions: define and manage information to include in her/his profile (e.g., birth date, age, sex, occupation, home city, friends, etc.), indicate her/his preferences about items (ratings), submit queries, and receive recommendations.

The user's preferences may be reflected in the system through explicit or implicit ratings. *Explicit ratings* can be expressed numerically (e.g., values in the range of one to five, zero to 10, etc.) or by using literal expressions (e.g., "good", "poor", etc.). Numeric ratings may be quite subjective (they depend heavily on the user's tendency to assign higher or lower scores, and therefore similarity metrics such as Z-score normalize the ratings to avoid this bias), but on the other hand literal expressions usually need to be translated to numeric values for the recommendation process. *Implicit ratings* are obtained through analysis of the user's behavior (e.g., purchases of specific products, visits to certain places, etc.). It is important to emphasize that the ratings assigned by the users may depend on the existing context when the rating was released. So, this contextual information is stored along with the ratings.

Moreover, the user can receive push-based recommendations and pull-based recommendations. Push-based recommendations are provided to the user based only on the user's profile and the current context, without a previous request by the user. On the other hand, pull-based recommendations are provided as an answer to a *query* explicitly submitted by the user (by using keywords related to the type of item required, by filling application forms and templates, by selecting among predefined queries, or by directly expressing the item type, for example, by choosing an option from a drop-down list). The user can enter, along with the query, *soft constraints* and *hard constraints* (e.g., the price of an item should be below a certain threshold, the item should not be located further than a certain distance, etc.) for the items as well as hints regarding the *context criteria* that should be considered for that type of item during the recommendation process.

3.2.2 Logic Layer

This layer contains the main modules of the system. A module called *Sentiment Analyzer* is envisioned to interpret an opinion in natural language (e.g., an opinion expressed in free text) and to determine if the opinion is positive, negative, o neutral, to assign it a rating automatically. It can apply techniques for *sentiment*

classification [LClC06, Liu10, Liu11, LZ12, BKKK16, YSZ17]. Methods of supervised learning, Naïve Bayes [DHS00], Support Vector Machines (SVMs) [VC95], and regression methods [KWPG01] can be used.

A module called *Rating Reliability Analyzer* would be in charge of protecting against profile injection attacks (attacks that try to artificially increase or decrease the relevance of some items by injecting fake ratings). This module would apply different attack detection methods [MBBW07] and its filtered output would be used to update the corresponding local knowledge base by using the module *Repository Manager*.

The *User Reliability Analyzer* module allows assigning more confidence to the information provided by the user's friends, by exploiting trust statements and user personal data in social networks [AMT05, MA07, MA15, LCCT17, XZZS17].

Pull-Based Recommendation Module

The *Pull-Based Recommendation* module is illustrated by the main steps shown below (see Figure 3.4):

- *STEP 1: Query.* First, the user introduces the item type as query in the system.
- *STEP 2: Context criteria and constraints.* The user can specify (optionally) hints regarding *context criteria* that should be taken into account during the recommendation process, as well as soft and hard constraints.
- *STEP 3: Context-aware recommendation paradigm.* Finally, a certain context-aware recommendation paradigm is considered to obtain the items to suggest.

In this module, the type of context-aware recommendation paradigm executed depends either on the final implementation of the architecture (provided by the system developer) and/or the type of items to recommend. The study performed in [PTG14b] provides a comparison of paradigms for context-aware recommendation in terms of the accuracy and diversity of the recommendations they provide, which is useful to determine which methods are more appropriate under certain circumstances.

With the *pre-filtering* paradigm, first a *context update* is performed. Then, a context analysis takes place during the *phase of obtaining contextualized data*. The contextualized data allows *creating/updating a 2D recommendation model*. Finally, the *item recommendations* are provided to the user. The *post-filtering* paradigm is similar to the pre-filtering but inverting the order of steps; as an example of a post-filtering step, filtering the candidate items according to specific location-dependent constraints [IMI10] could be considered (e.g., filtering the items based on their distance from the user). Finally, in the *contextual modeling* paradigm, the context information is used directly in the recommendation model. In this case, the recommendation model can be a classification technique, where the context variables and the attributes of items are considered features and the ratings are the target classes.

It should be noted that in a mobile environment, where the context is constantly changing, the recommendations should be kept up-to-date over time. For that purpose, the recommendation modules include a loop. For example, both the *context*

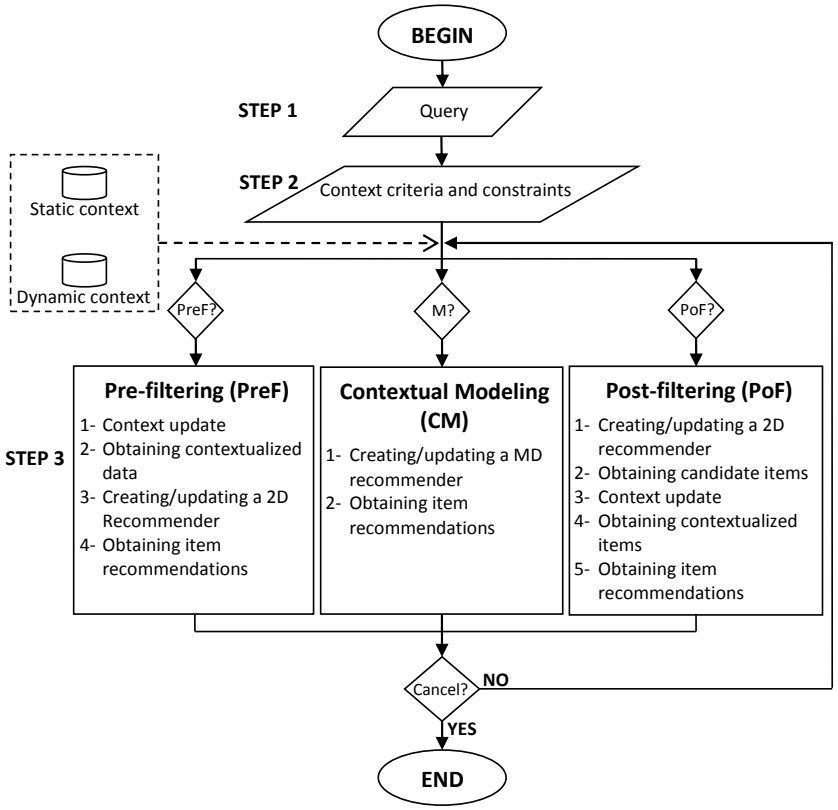


Figure 3.4: Main steps of the pull-based recommendation module.

update and *obtaining contextualized data* tasks must be executed periodically, with a certain frequency.

Push-Based Recommendation Module

On the other hand, the *Push-Based Recommendation* module is based on the pre-filtering and post-filtering paradigms, adapted for mobile environments by including a continuous reevaluation of context updates (see Figure 3.5). Below, we show the main steps for a push-based recommendation:

- *STEP 1: Context update.* First, the context is automatically captured from the user's environment.

- *STEP 2: Recommendation triggering.* The user's contextual conditions are analyzed to decide when to trigger a recommendation process.

- *STEP 3: Context-aware recommendation paradigm.* Finally, if the MD recommendation algorithm is executed, then the pre-filtering and post-filtering recommendation paradigms are considered to obtain the items to suggest.

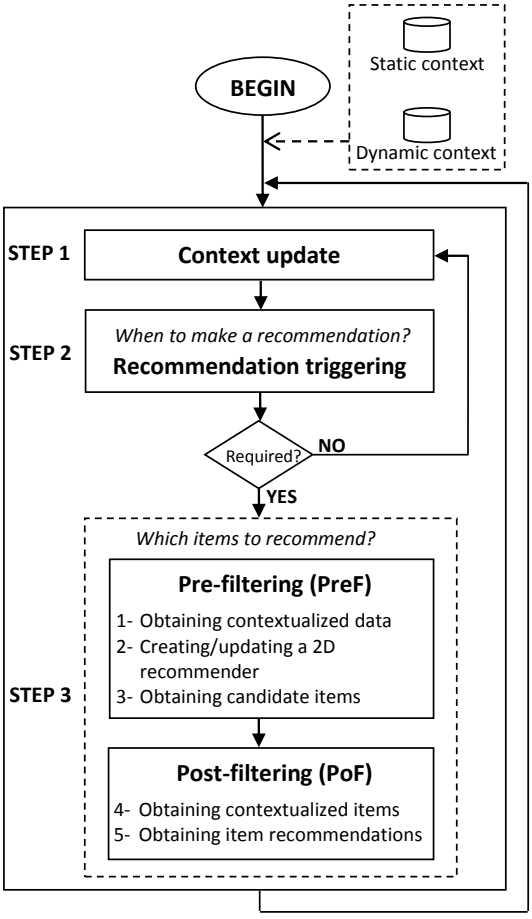


Figure 3.5: Main steps of the push-based recommendation module.

Firstly, the *context update* and *recommendation triggering* tasks are performed in order to detect when the system should provide a recommendation to the user. For this analysis, the system uses the contextual conditions automatically detected from the user's environment. If a recommendation should be pushed to the user, then the pre-filtering and post-filtering recommendation paradigms are performed. The pre-filtering algorithm filters the appropriate data for the current context, and then a 2D recommender is executed to obtain a set of candidate items to suggest. Finally, the resulting list of items can be provided to the user by applying a post-filtering,

which solves the possible conflicts between candidate items (e.g., this phase may filter candidate restaurants to recommend by taking into account suggested user's private preferences, such as the need of gluten-free meals, or could adjust the ranking of activities in an appropriate order).

3.2.3 Data Layer

This layer abstracts us from the details of access to the data, which can be stored in relational databases, plain files, etc. The module *User Profile and Context Manager* is responsible for managing (inserting, modifying, and removing) information of the user's profile and context. The information can be stored in a local knowledge base through the module *Repository Manager*, which allows the access to the local knowledge base. For example, if the data are stored in a relational database, it is possible to access relevant data (e.g., about users, items, contexts and ratings) for the recommendation process without the need to directly apply SQL queries. The information needed to perform the recommendation process can be stored in a centralized database or in local databases (located on the mobile devices of the users). In this last case, the mobile device of a user can exchange data with others in a P2P way.

Context Model

The context model that we propose, inspired by [MP13], is shown in Figure 3.6. The *user_item_context_rating* entity represents a context-aware recommendation process, which includes the dimensions $U \times I \times C \rightarrow R$. It has exactly one *User*, exactly one *Item*, but zero or more *Context* entities. Hence, a user can rate several items in none or different contexts. If there is no context information, then the recommendation process can be reduced to a traditional 2D algorithm. The *ratings* are integer values in a specific range. However, the user can also give *opinions* in natural language, which can be converted into a numerical value (or rating) by using the *Sentiment Analyzer* module. The *user_provided* property specifies if the rating is manually obtained by the user or automatically generated from a user's textual opinion.

The *Context* entity represents a set of *context variables* with their corresponding *types* (e.g., user context, item context, environment context, system context, etc.), *names* (e.g., mood, location, season, weather, etc.), and *values* (happy, near, summer, hot, etc.). In order to facilitate calculations of soft context similarities, we explicitly include a reflexive relation between *context variables*. We also consider in the model whether the context variable is *dynamic* or static. The *type* property, from the perspective of context-aware computing, can be optionally related with the *Item* and *User* entities (e.g., user context and item context). The *Item* entity, may have a set of properties such as the *name* and *location*. Besides, this entity can be classified in zero or one *type* of item (e.g., restaurant, music, book, museum, etc.), as well as have one or more *features* that describe it. Moreover, the *User* entity may have associated *profiles*, and each of these profiles may have *variables* that characterize them.

The extension of the context model [MP13], proposed by us and shown in Figure 3.6, implies a set of important differences, such as the following:

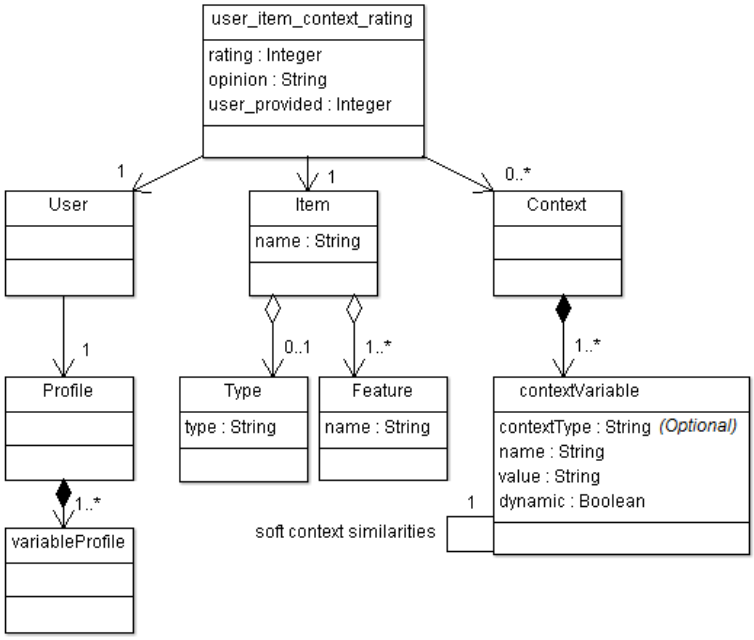


Figure 3.6: The proposed context model for CARS, inspired by [MP13].

- We do not represent the *rating* property in the *Context* entity, since we consider that the same context may have ratings for different users.
- We also allow *opinions* in natural language, from which the ratings can be inferred.
- We allow the computation of soft context similarities, by considering a reflexive relation between the *context variables*.
- We also allow context variables that are not associated with any predefined *context type* (e.g., user context, item context, environment context, system context, etc.). So, the type of context associated with a context variable is optional.
- We allow the use of the model for cases where there is no associated context information.
- We allow one *user* with exactly one *profile* representing her/his preferences. For example, during the recommendation process of an item, the distance (between the location of the user and the item) may be relevant according to the user's preferences.

Finally, it is interesting to mention that, in addition to the data related to the context model, this layer can also provide other types of information, such as data about roads and maps of buildings.

Data Sharing Manager Module

Obtaining data from external knowledge bases is also possible thanks to the *Data Sharing Manager*. In this module, we advocate mobile P2P context-aware recommendations that exploit exclusively short-range wireless ad hoc communications (e.g., Wi-Fi), which are usually assumed to provide a communication range of 200-300 meters, to exchange data among mobile devices that are within the communication range of each other [dCRHITH17]. In this way, there is no need of a fixed support infrastructure and no centralized server is devoted to collecting all the data (e.g., ratings) provided by the users. Instead, the data are propagated opportunistically (i.e., as the devices meet when they move around the space) through the ad hoc network and the votes provided by the users are stored on the mobile devices of the users in a distributed way. Then, the recommendation system running locally on each mobile device can exploit its local database to predict ratings and recommend items to its user.

Mobile P2P architectures offer several potential advantages over centralized solutions (see Section 2.1.1). For example, the following advantages can be mentioned: they are costless in terms of the infrastructure required (deploying a fixed support infrastructure may be expensive); the mobile users do not incur any cost derived from the use of cellular communications (e.g., 3G or 4G) when providing rating information; and they may provide better privacy guarantees, as no centralized server collects all the information provided and the transmission of data is usually confined to nearby spatial regions. So, a P2P approach could provide a number of benefits, as it does not require a fixed support infrastructure that collects all the information, and besides it can always be complemented (if needed) by some fixed support nodes.

More precisely, when a user releases a rating (i.e., provides a vote with her/his assessment of an item), the rating information (user id, item id, context data, and rating provided) is broadcast in its vicinity: all the mobile devices within its communication range listen to the information and store it in their local databases. Thus, the rating information propagates through the mobile devices of the users according to their spatio-temporal relevance: the rating information that has been released recently will reach first users located nearby. The *time needed to propagate* data by broadcasting it in the wireless medium (TTP) is determined by the following equation:

$$TTP = l + \frac{N}{v} \tag{3.1}$$

where l is the latency, N is the size of the data to transfer, and v is the communication bandwidth available.

The data dissemination approach used for the propagation of rating information through the mobile ad hoc network, once it has been released for the first time, is based on *contention-based forwarding*. This implies that, instead of using a flooding approach, which is known to be subject to a number of problems (mainly, the storm broadcasting problem [NTCS99], that may cause a major network overload as well as interferences due to network packet collisions), only a single mobile user will be in charge of propagating each specific rating by broadcasting it to all her/his neigh-

bors. All the mobile devices within the communication range listen to the wireless medium and receive the rating information broadcast, but only one device at a time is responsible for forwarding a specific piece of rating information to other nodes. In this way, the protocol limits the amount of retransmissions of a single piece of data. As suggested by other studies [CDI11, IDTL15], the mobile device of the user located at a furthest distance from the last relay is chosen, in order to maximize the probability that the rating information will travel a large spatial distance as quickly as possible. Indeed, that specific mobile device is expected to have the greatest number of neighbor devices not yet informed about the rating information being transmitted. As an example, Figure 3.7 illustrates a couple of communication chains.

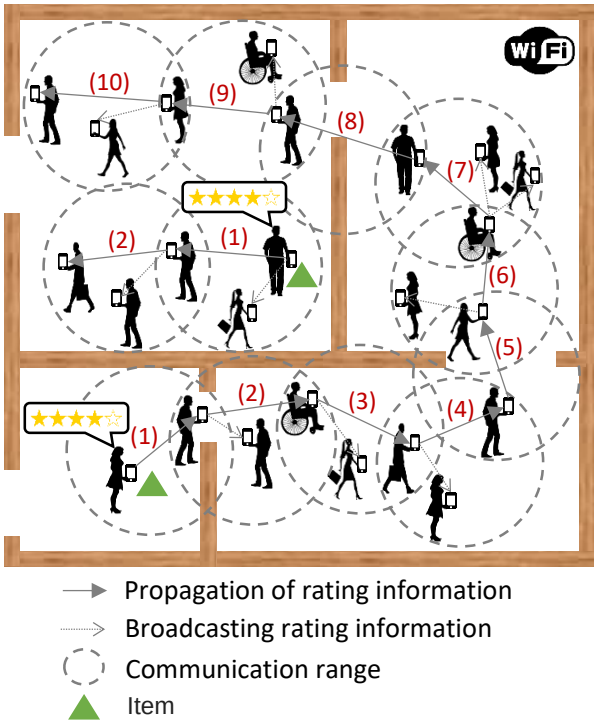


Figure 3.7: Mobile P2P data propagation.

Moreover, we also allocate a specific *Time-to-Live* (*TTL*) to each piece of data (e.g., 3 minutes), which controls how long it will be kept alive in the mobile P2P network: when the TTL expires, that rating information is discarded by the current forwarder and not propagated anymore through the network (i.e., it will not be transmitted to other mobile devices). In this way, by adjusting the TTL value, we can avoid old ratings (released in out-of-date contexts) from further propagation. For each piece of rating information, as long as the TTL is greater than 0, the forwarding responsibility is thus handed over to the furthest mobile device within the commu-

nication range. In case there is no neighbor within the communication range, the forwarding role remains assigned to the same mobile device, which will retransmit that piece of data with a certain *retransmission period* until a new data forwarder can be assigned. The selection of the furthest forwarded is based on the use of *backoff timers* whose duration is inversely proportional to the distance from the last relay. More details about these types of contention-based dissemination protocols can be found in works such as [CDI11, IDTL15].

3.3 Summary of the Chapter

In this chapter, we firstly presented two motivating scenarios to illustrate the interest of a context-aware mobile recommendation architecture. Secondly, we described MOONRISE, an architecture for context-aware recommendations in mobile environments. The architecture is generic, extensible and can be adapted to the requirements of specific types of recommendations. Specifically, we presented the different layers of the architecture, detailing the corresponding modules per layer. We also explained the main steps of the pull-based recommendation module, which accommodates the pre-filtering, post-filtering and contextual paradigms. In an analogous way, we explained the push-based recommendation module. Both models (pull-based and push-based recommendations) are considered as the main modules of the architecture. In the rest of this thesis, we will focus on the description of the main modules related to the logic layer, such as the *Pull-Based Recommendation* module (Section 4.1), and the *Push-Based Recommendation* module (Section 4.2). The main goal of this chapter was to show the interest and justification of the architecture designed. Our proposal is a first step to contribute to bridging the gap between context-aware recommendation systems and mobile computing.

Chapter 4

Context-Aware Mobile Recommendation Approaches

In this chapter, we focus on the main modules of the proposed architecture. In Section 4.1, we present an overview of the pull-based recommendation module implemented in the architecture. We describe the context-aware recommendation paradigms used in the pull-based recommendation process, which consider aspects related to the mobile environment. We also explain how to compare (in the pre-filtering and post-filtering paradigms used) the current user's context and the contexts stored in the knowledge base, by using a context similarity metric that we propose. In Section 4.2, we describe a push-based recommendation model for mobile users, which takes into account the context of recommendations and the surrounding events. For illustration, we also describe a case study that shows the interest and feasibility of the push-based recommendation proposal. Besides, in Section 4.3, we present an example of a trajectory-based recommendation approach that pushes suitable recommendations to mobile users.

4.1 Pull-Based Recommendation Approach

In this section, we present the *Pull-Based Recommendation* module [dCRHI16]. This module provides explicit (or reactive) recommendations, obtained as an answer to a query explicitly submitted by the user and evaluated by the system as a continuous query [TGNO92].

4.1.1 Overview of the Pull-Based Recommendation Process

In Chapter 3, we presented a high-level view of the pull-based recommendation process, where the user first introduces the *item type* (as the query) and *context constraints* (hard and/or soft constraints can be considered) in the system. Afterwards, the user might (optionally) specify hints regarding the importance of the different

context criteria that should be taken into account during the recommendation process. Finally, a certain context-aware recommendation paradigm (*pre-filtering*, *post-filtering*, or *contextual modeling*) is applied to obtain appropriate recommendations for the user. The type of paradigm executed depends of the choice of the system developer and/or the type of items to recommend. As the context will continuously change (e.g., imagine a scenario where both the user and the items to recommend move, such as a taxi recommendation scenario), the recommendation system must automatically update the contextual information (context of the user and context of the items) and evaluate the query in a continuous way. This continuous revaluation process will be performed until the user decides to cancel the query (i.e., when she/he is not interested in recommendations about that type of item anymore).

Algorithm 1 shows the pull-based recommendation process in more detail. It receives as input: the contextual information of the current user, which is obtained from mobile sensors and/or the user himself/herself; the kind of items required; optionally a maximum number of items k to recommend; the minimum score required for two contexts to be considered similar (called the *similarity threshold*); and the time interval between two consecutive recomputations of the recommendation (the *refreshment period*). The sub-algorithm *contextAwareMobileRecommendation*, can be implemented according to any context-aware recommendation paradigm: pre-filtering, post-filtering, and contextual modeling; the details of these paradigms are provided in sections 4.1.3, 4.1.4, and 4.1.5, respectively. This algorithm is executed with the required refreshment frequency until the query is explicitly cancelled by the user. For this purpose, the values of the dynamic context variables are updated in each refreshment.

Algorithm 1: Pull-based recommendation algorithm

Input: A set of context variables ($C = \{c_1, c_2, \ldots, c_n\}$), the type of items required (*type*), optionally the maximum number of items to recommend (k), the minimum context similarity required ($simThreshold$), and the refreshment period ($refreshPeriod$). It is assumed that *continue* (initially set to true) will be set to false when the user decides to cancel the recommendation process.

Output: Continually keeps the set of items to recommend up-to-date ($items = \{i_1, i_2, \ldots, i_m\}$, with $m \leq k$ in case k is provided).

```
1 items ← ∅;
2 while (continue == true) do
3 |   initialTime ← getCurrentTime();
4 |   items ←
  |     contextAwareMobileRecommendation(C, type, k, simThreshold, items);
5 |   timeToSleep ← refreshPeriod − (getCurrentTime() − initialTime);
6 |   if timeToSleep > 0 then
7 |   |   sleep(timeToSleep);
```

4.1.2 Comparing Contexts

Both the pre-filtering and the post-filtering paradigms require determining the similarity between contexts, as the recommendation model will need to identify past votes/ratings (from the same user and from other users) that have a context similar to the current context of the user. In this section, we first describe the representation of contexts using context variables and then introduce a context similarity metric.

Context Variables

When comparing two contexts, the different context variables composing them have to be compared. In principle, all the context variables can be considered as equally important; so, we could assume that each context variable has a weight that is obtained by dividing one between the number of context variables. However, there are circumstances where some variables are more important than others, which should be considered during the recommendation process. Hence, for each domain (i.e., type of items) the recommendation system designed should assume by default a higher *importance weight* for the context variables that are usually considered more relevant and a value of zero for those that are irrelevant in that domain. Moreover, the user can optionally modify her/his *context criteria*, by adjusting one or all the weights of the variables. For example, the user could consider some context variables as irrelevant, assigning them a weight of zero. Alternatively, she/he could highlight the importance of some context variables over others, assigning them higher weights. If the user modifies only one or a few weights, then the values of the remaining weights will be automatically adjusted (to keep their desired relative importance and still ensure that the sum of all the weights is equal to one). These weights are considered during the recommendation process.

Example:
Let us suppose that a restaurant recommendation system considers that the variables more relevant in that domain are: companion, weekday, transport way, mobility, time of the day, mood, temperature, weather, and price. For these variables, the system automatically assigns default relevance weights that are assumed to be generally suitable for that type of items, ensuring that the sum of all the weights is 1 (see Table 4.1). However, the user U_1 considers that the variables that are relevant to her/him are: transport way, companion, price, time of the day, and weather; hence, she/he modifies all the initial weights. On the other hand, the user U_2 considers that the most important context variables are others (transport way, price, and time of the day), and so decides to assign other weights.

	transport way	**mobility**	**weekday**	**mood**	**companion**	**price**	**season**	**temperature**	**time of the day**	**weather**
Initial:	0.1	0.1	0.1	0.1	0.1	0.2	0.0	1.0	0.1	0.1
U_1:	0.3	0.0	0.0	0.0	0.1	0.3	0.0	0.0	0.2	0.1
U_2:	0.3	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.3	0.0

Table 4.1: Example of weight vector for the context variables.

Context variables whose weights are different from zero represent *soft constraints*, as they embed preferences of the user regarding the impact of the different context variables. Moreover, the user could also formulate *hard constraints*, which are specific conditions on the values of certain context variables that need to be satisfied (e.g., the price of an item should be below a certain threshold, the item should not be located further than a certain distance, etc.).

Context Similarity Metric

In Algorithm 2, we present the strategy used to calculate the similarity between context variable vectors C_x (i.e., the context of the current user) and C_y (i.e., the context of another user or the context of an item), of length n, by using Equation 4.1. As shown in the equation, the similarity between contexts is computed based on the similarity between the values for each of the context variables (e.g., transport way, mobility, weekday, mood, companion, price, season, temperature, time of the day, weather, etc.). Specifically, the similarity $sim(c_{x_i}, c_{y_i})$ computed between each pair of values c_{x_i} and c_{y_i} (of the context variable i of the contexts c_x and c_y, respectively) is multiplied by the relevance *weight* w_i corresponding to that context variable. If the overall similarity is greater than a certain *similarity threshold*, then the contexts C_x and C_y are considered to be similar. The proposed equation is analogous to the one used in [LCVA12]. However, in that work only the similarity between the context of a current user (C_x) and an item context (C_{photo}, as that work focuses in the recommendation of photos) was considered, whereas we consider also the possibility to compare two user's contexts.

$$similarity\,(C_x, C_y) = \sum_{i=1}^{n} sim(c_{x_i}, c_{y_i}) * w_i \tag{4.1}$$

To determine the similarity $sim(c_{x_i}, c_{y_i})$ between the values of a context variable i in two contexts, we consider the following specific cases:

- If the values of the same context variable in the two contexts are identical (i.e., $c_{x_i} = c_{y_i}$), then their similarity is 1. For example, in Table 4.2, $c_{x3} = c_{y3}$, $c_{x5} = c_{y5}$, and $c_{x7} = c_{y7}$, for the context variables *weekday*, *companion*, and *season*, respectively.

- If c_{xi} and c_{yi} are considered to be completely different, then their similarity is 0. According to the example in Table 4.2, the context variables *transport way*, *mobility*, *mood*, *price* and *time of day* are examples of this (e.g., $c_{x1} \neq c_{y1}$, $c_{x2} \neq c_{y2}$, $c_{x4} \neq c_{y4}$, $c_{x6} \neq c_{y6}$, and $c_{x9} \neq c_{y9}$).

- Finally, if the values of the context variable are not exactly the same but have a close semantic relationship (*areRelated()* in Algorithm 2), then the similarity takes some value between 0 and 1 (e.g., 0.5). This value (*softSimilarity()* in Algorithm 2) has to be defined and stored previously and it represents a way to explicitly encode soft distances between possible values of a variable. For

Algorithm 2: CONTEXT SIMILARITY

Input: Two sets of context variables ($C_x = \{c_{x1}, c_{x2}, \ldots, c_{xn}\}$ and $C_y = \{c_{y1}, c_{y2}, \ldots, c_{yn}\}$), the minimum similarity required ($simThreshold$), and the weights for the different context variables ($weights = \{w_1, w_2, \ldots, w_n\}$).

Output: A boolean indicating whether the two contexts provided are considered to be *similar*.

```
similar = false;
similarityScore ← 0;
for i ← 0 to n do
    if (IS_NULL(c_xi) ∨ IS_NULL(c_yi)) then
        similarityBetweenVariables ←
          compareVariablesWithMissingInfo(c_xi, c_yi);
    else
        if (c_xi == c_yi) then
            similarityBetweenVariables ← 1;
        else
            if areRelated(c_xi, c_yi) then
                similarityBetweenVariables ← softSimilarity(c_xi, c_yi);
            else
                similarityBetweenVariables ← 0;
    similarityScore ← similarityScore + similarityBetweenVariables * w_i;
if similarityScore ≥ simThreshold then
    similar ← true;
return similar;
```

	1	2	3	4	5	6	7	8	9	10
	transport way	**mobility**	**weekday**	**mood**	**companion**	**price**	**season**	**temperature**	**time of the day**	**weather**
x	car	moving	weekend	happy	alone	expensive	summer	hot	morning	clear sky
y	walking	stopped	weekend	sad	alone	free	summer	warm	afternoon	sunny

Table 4.2: Example of two context variable vectors of users: dense vectors.

example, for the context variable *temperature* the distance between "hot" and "warm" (c_{x8} and c_{y8} in Table 4.2) is smaller than the distance between "hot" and "cold", and regarding the *weather* the distance between "cloudy" and "rain" is smaller than the distance between "rain" and "sunny". Another example can be illustrated by the context variable *weather* in Table 4.2, where the values *clear sky* and *sunny* can be considered to be semantically close.

Comparing Sparse Contexts

In many cases, it may happen that a complete description of a context is not available. In other words, we could have context variables that have an unknown or null value. Directly applying the method described above to compare sparse context vectors may lead to similarity scores that are not appropriate. To deal with missing information, we have included in Algorithm 2 the method *compareVariablesWithMissingInfo*, which can consider several strategies:

- Regarding the comparison between an unknown value of a context variable with another value, we could assume *minimum similarity* (i.e., a value of 0, which corresponds to the pessimistic assumption that the real value missing would be different from the other one), *maximum similarity* (i.e., a value of 1, which corresponds to the optimistic assumption that if we knew the value missing this would be equal to the other one), or *neutral/medium similarity* (e.g., a value of 0.5).

  For instance, in Table 4.3 one of the values of the user's context variable *mobility* (c_{z2}) is unknown and the other one has the value *moving* (c_{w2}). An optimistic comparison implies considering that the other user was also moving (maximum similarity), whereas a pessimistic comparison leads to the assumption that the other user was probably in a static location. A more neutral approach leads to consider a similarity of 0.5 as an average of the similarity values computed for the two other cases (1 and 0, respectively). Similarly, in Table 4.3 one of the values for the context variable *companion* is unknown (c_{w5}) and the other one has the value *alone* (c_{z5}), so a similar reasoning can be applied.

	1	2	3	4	5	6	7	8	9	10
	transport way	**mobility**	**weekday**	**mood**	**companion**	**price**	**season**	**temperature**	**time of the day**	**weather**
z	unknown	unknown	weekend	happy	alone	expensive	summer	hot	morning	clear sky
w	unknown	moving	weekend	sad	unknown	budget traveler	summer	warm	afternoon	sunny

Table 4.3: Example of two context variable vectors of users: sparse vectors.

- If the two values of a context variable i are both unknown, two strategies are possible. On the one hand, we can decide to *ignore the comparison between the two unknowns*, which means that the context variable i is not taken into account. On the other hand, it is also possible to *perform the comparison between the two unknowns* anyway, by applying one of the strategies mentioned above (that is, assume minimum similarity, consider maximum similarity, or estimate neutral/medium similarity).

  As an example, in Table 4.3, for the context variable *transport way* the values for both users are unknown. The weights for variables whose values are unknown in the two contexts to compare are assumed to be zero and the rest of the weights are adjusted dynamically to make the overall sum of weights equal to 1 (unless all the other weights are also zero, in which case no adjustment is possible and one of the alternative strategies described in the following should be applied).

So, we can ignore that context variable. Alternatively, we can assume that they were actually using the same transportation means (maximum similarity), using a different one (minimum similarity), or using a similar one (medium similarity). Using a medium similarity can also be interpreted from a statistical perspective: in this case, it is equally likely that the values match or do not match, so an overall average similarity can be considered.

4.1.3 Pre-filtering Paradigm

With the *pre-filtering* paradigm (see Algorithm 3), first a *context update* is performed (i.e., the information regarding the context of the user is updated, by using sensors or explicit cues provided by the user). Then, a *context analysis* takes place to provide *contextualized data* by determining the similarity between the current context of the user C_{cu} and other user contexts C_j included in past ratings, based on the use of Algorithm 2. The contextualized data is then provided as an input to a traditional recommendation algorithm. So, we could say that with this process we remove irrelevant raw data (i.e., data that are not related to the current context of the user) and that the data selected as relevant (*smart data*) are used to build the recommendation model. Finally, a maximum of k *items* are recommended to the current user.

Algorithm 3: PREFILTERING paradigm

Input: A set of context variables of the current context of the user ($C_{cu} = \{c_{cu1}, c_{cu2}, \ldots, c_{cun}\}$), the type of items required ($type$), optionally a maximum number of items to recommend (k), the minimum context similarity required ($simThreshold$), and information about past votes provided by the users u about items i ($ratings = \{(user_u, item_i, rating_1, C_1), (user_u, item_i, rating_2, C_2) \ldots, (user_u, item_i, rating_h, C_h)\}$).

Output: A set of $items = \{item_1, item_2, \ldots, item_m\}$ to recommend (with $m \leq k$, in case k is provided).

```
1 ratingsFiltered ← ∅;
  // for each rating available
2 for j ← 1 to h do
3     if (item_j.type == type) then
4         similar ← contextSimilarity(C_j, C_cu, simThreshold);
5         if (similar == true) then
6             ratingsFiltered ← ratingsFiltered ∪ {(user_j, item_j, rating_j, C_j)};
7 recommender ← createTraditionalRecommender(ratingsFiltered);
8 items ← recommender.recommend(C_cu, k);
9 return items;
```

4.1.4 Post-filtering Paradigm

The post-filtering paradigm (see Algorithm 4) is similar to the *pre-filtering* paradigm, but inverting the order of the steps. First, it applies a traditional recommendation model to obtain *candidate items* and later *contextualized candidate items* are provided to the current user. Moreover, the post-filtering algorithm considers hard constraints to filter out items that do not satisfy those constraints. Notice that the pre-filtering algorithm proposed (presented in Section 4.1.3) does not check hard constraints to filter the items that will be used to learn the model; although hard constraints could also be considered in the pre-filtering, that would decrease considerably the amount of items used for training the model.

Algorithm 4: POSTFILTERING paradigm

Input: A set of context variables of the current context of the user ($C_{cu} = \{c_{cu1}, c_{cu2}, \ldots, c_{cun}\}$), the type of items required ($type$), optionally a maximum number of items to recommend (k), the minimum context similarity required ($simThreshold$), information about past votes provided by the users u about items i ($ratings = \{(user_u, item_i, rating_1, C_1), (user_u, item_i, rating_2, C_2) \ldots, (user_u, item_i, rating_h, C_h)\}$), and the strict constraints required ($hardConstraints$).

Output: A set of $items = \{item_1, item_2, \ldots, item_m\}$ to recommend (with $m \leq k$, in case k is provided).

```
1 recommender ← createTraditionalRecommender(ratings);
2 candidateItems ← recommender.recommend(C_cu);
3 items ← ∅;
4 if (hardConstraints ≠ ∅) then
      // Algorithm 5
5     candidateItems ← filteringWithHardConstraints(C_cu, type,
        candidateItems, hardConstraints);
  // Algorithm 6
6 items ← filteringWithSoftConstraints(C_cu, type, candidateItems,
    simThreshold);
7 sort(items);
  // in decreasing order of the predicted rating and remove all but
     the first k items
8 return items;
```

As shown in Algorithm 4, to obtain these contextualized items, both hard constraints and soft constraints could be applied. The application of hard constraints represents a strict requirement regarding the fulfillment of certain conditions, and therefore only the items that match all the hard constraints of the current user will be recommended (see Algorithm 5); as an example, filtering the candidate items according to specific location-dependent constraints [IMI10] could be considered (e.g.,

filtering the items based on their distance from the user). The application of soft constraints implies a traditional comparison between the context of the user and the context of each candidate item (see Algorithm 6).

Algorithm 5: FILTERING WITH HARD CONSTRAINTS

Input: The constraints to satisfy ($hardConstraints = \{constrainst_1, constrainst_2, \ldots, constrainst_l\}$), the initial set of candidate items ($candidateItems = \{item_1, item_2, \ldots, item_o\}$), the current context of the user (C_{cu}), and the type of items required ($type$).

Output: A set of items to recommend ($itemsToRecommend = \{item_1, item_2, \ldots, item_m\}$).

```
1 itemsToRecommend ← ∅;
2 for j ← 0 to o do
3     if (item_j.type == type) then
4         satisfied ← satisfyAllHardConstraints(item_j, C_cu,
            hardConstraints);
5         if (satisfied == true) then
6             itemsToRecommend ← itemsToRecommend ∪ {item_j};
7 return itemsToRecommend;
```

Algorithm 6: FILTERING WITH SOFT CONSTRAINTS

Input: The initial set of candidate items ($candidateItems = \{item_1, item_2, \ldots, item_o\}$), the current context of the user (C_{cu}), the type of items required ($type$), and the minimum similarity required ($simThreshold$).

Output: A set of items to recommend ($itemsToRecommend = \{item_1, item_2, \ldots, item_m\}$).

```
1 itemsToRecommend ← ∅;
2 for j ← 0 to o do
3     if (item_j.type == type) then
4         similar ← contextSimilarity(item_j, C_cu, simThreshold);
5         if (similar == true) then
6             itemsToRecommend ← itemsToRecommend ∪ {item_j};
7 return itemsToRecommend;
```

4.1.5 Contextual Modeling Paradigm

Finally, in the contextual modeling paradigm (see Algorithm 7) the contextual information is used directly in the recommendation model, as context variables are simply considered as features in the feature vectors compared. Specifically, in our

prototype we use a Naïve Bayes classifier [JL95], which is a probabilistic classifier based on Bayes' theorem, but other approaches could be used, such as a SVM classifier [VC95] (e.g., see [ONMU06, XA06]) or association rule mining [RA+94] (e.g., see [SMB07, GK17]).

Algorithm 7: CONTEXTUAL MODELING paradigm

Input: The minimum score needed to recommend an item ($recommendationThreshold$), the profile of the user ($userProfile = \{(C_1, item_1, rating_1), (C_2, item_2, rating_2) \ldots, (c_h, item_h, rating_h)\}$), the potential items to recommend along with their context ($potentialItems, itemContexts$), the type of items required ($type$), and the current context of the user (C_{cu}).

Output: A set of items to recommend ($itemsToRecommend = \{item_1, item_2, \ldots, item_m\}$).

```
1  userKnowledgeBase ← ∅;
2  for j ← 0 to h do
3      features(j) ← {(C_j, item_j) | (C_j, item_j, rating_j) ∈ userProfile};
4      class ← rating_j;
5      userKnowledgeBase ← userKnowledgeBase ∪ {(features, class)};
6  classifier ← createClassifier(userKnowledgeBase);
7  candidateItems ← ∅;
8  for j ← 0 to size(potentialItems) do
9      if (potentialItems(j).type == type) then
10         instance ← (itemContexts(j), potentialItems(j));
11         ratingPredicted ← classifier.classify(instance, C_cu);
12         candidateItems ← candidateItems ∪ {(instance, ratingPredicted)};
13 sort(candidateItems);
   // in decreasing order of the predicted rating
14 itemsToRecommend ← ∅;
15 for j ← 0 to size(candidateItems) do
16     if ((candidateItems(j).ratingPredicted ≥
       recommendationThreshold) ∧ (itemsToRecommend.size() ≤ k) then
17         itemsToRecommend ← itemsToRecommend ∪ {candidateItems(j)};
18 return itemsToRecommend;
```

In the contextual modeling paradigm, the classifier learns a model for each user profile. The model is stored in a knowledge base, which contains several instances (one for each user). Each instance is composed by the following *features or attributes*: the context variables (of nominal type) and the item's rating for those contexts (numeric type). The *decision class* (nominal type) can be the different discrete values of the ratings (e.g., integer values in a scale between one and five). For a new instance (context variables of the current user and a possible item to recommend) the classifier

predicts a rating. The ratings predicted for the different candidate items are sorted (in descending order of rating) and the first k items with a predicted rating higher than a specific *recommendation threshold* (e.g., the items with a rating predicted of at least 4) are provided to the user.

Another variant for this paradigm would be to consider as the decision class the labels *like* or *dislike*, by partitioning the possible rating values in these two categories (e.g., the ratings 1, 2 and 3 could belong to the *dislike* class –i.e., items that should not be recommended– and the ratings 4 and 5 to the *like* class –i.e., items that should be recommended–). In this case, that implies approaching the recommendation problem as a binary classification problem, the algorithm would recommend the k *items* with the highest probability to belong to the class *like*.

4.1.6 Supporting Pull-Based Recommendations with a Keyword-Based Searching Approach

In this section, we describe two possible solutions to the problem of identification of the type of item (e.g., music, movie, book, etc.) that the user specifies, by using keywords, in a pull-based recommendation system [dCRHITLG16]. For example, if a user introduces in the system the keywords "place to eat" the system must be able to interpret that the user is searching items of the type "restaurant" without the need to choose the item type from a list previously defined by the system.

An alternative proposed is based on the use of the Hidden Markov Model (HMM) [Rab89, Edd96, Fin14] and another one relies on the exploitation of Information Retrieval models [SM86, FBY92, BYRN99, MRS08]. Both solutions are implemented in the architecture proposed. According to the experimental results presented in Appendix C, the HMM-based approach performs generally better than the IR-based approach in terms of accuracy. However, a potential problem with the HMM model is how to determine suitable probability values. Besides, it would be interesting to analyze how the use of query expansion techniques (e.g., thesaurus) or semantic relations between words (mainly synonyms) could improve the experimental results obtained (in particular, for example, if the user's input includes keywords that are not present in the available datasets).

It should be noted that in the field of recommendation systems, few works are related to keyword-based searching. For example, in [SZ11] the authors studied two methods of recommendation systems for personalizing and improving the search results, by using the collaborative users' knowledge and integrating the information from the users social network. For the movie recommendation system proposed in [SNC13], a hybrid system (combining the collaborative filtering and the content-based recommendation techniques) that alleviates the noise and semantic ambiguity problems from the user-generated content (e.g., keyword and tag representations). However, both works are focused only on searching on unstructured data sources. Moreover, the research presented in [SS15], where the authors proposed the use of keywords to indicate users' preferences from a keyword candidate list, to generate appropriate recommendations based on a hybrid filtering algorithm, is focused only on improving

the scalability and efficiency of a Big Data environment. The lack of keyword-based approaches specifically designed for recommendation systems motivates the proposals that we present in this section.

4.1.6.1 HMM Approach

Let us define a Hidden Markov Model λ as a triple A, B, π, where:

- $A = a_{ij}$ are the state transition probabilities.
- $B = [b_j(T)]$ are the observation probabilities.
- $\pi = [\pi_i]$ are the initial state probabilities.

There are mainly two basic problems associated to an HMM [Rab89]:

1. Problem 1. Given the observation sequence $O = O_1, O_2, \ldots, O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1, q_2, \ldots, q_T$ with the highest probability $P(Q|O, \lambda)$, that is, the one that best "explains" the observations?
2. Problem 2. How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

According to the existing literature, the first problem (i.e., finding the most likely explanation for an observation sequence) can be efficiently solved by using the Viterbi algorithm [GDF73, Lou95]. Adapting that algorithm to be used in our system, we have to define the following structures (the model λ is composed of A, B and π):

- Q is the set of states, which will be composed of the feature names (that characterize the items) and the item type.
- O is the set of observations, which will be the item types, as well the names and values of the item features.

As an example, we show in Figure 4.1 a fragment of the HMM proposed for a dataset InCarMusic [BKL+11]. The idea is the same for any other dataset.

The Viterbi algorithm needs as input the observations (B) and the HMM model (λ). These values are obtained automatically from the database. An observation file contains the values and the names of the item features (e.g., title, artist and category in the case of Figure 4.1) and the item types (e.g., music in the case of Figure 4.1). Moreover, another file describes the HMM model. Considering the example of Figure 4.1, the structure of the file for three states (e.g., music-title, music-artist, and music-category) and several observations (e.g., title, t_1, t_2, artist, a_1, a_2, a_3, category, c_1, c_2, and music) is displayed in Figure 4.2.

In the model λ, each state contains the state transition probabilities A, the observation probabilities B, and the initial state probabilities π. By default, the probability

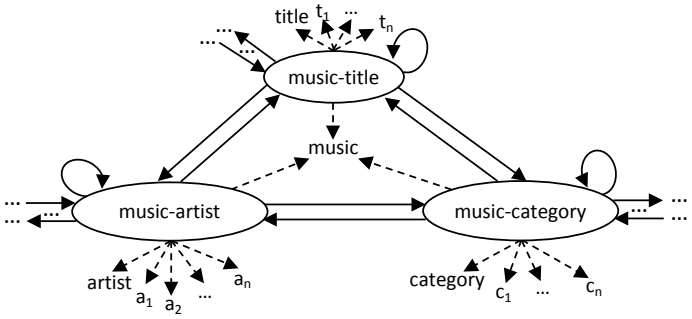


Figure 4.1: Representation of the HMM model for the InCarMusic database.

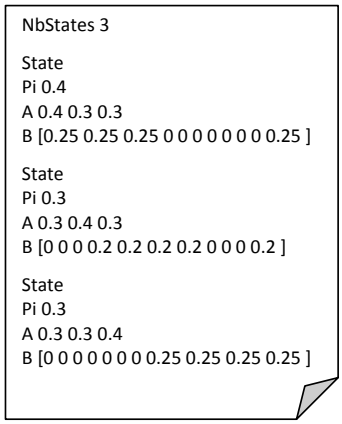


Figure 4.2: Example of the structure of an HMM model stored.

values by state of the vector B are equally distributed on all the observations, dividing one by the number of terms related to the current state (an example is shown in Figure 4.2). The state transition probabilities A have the same values for all the states, obtained by dividing one by the number of states. The initial state probabilities π are determined similarly. Nevertheless, our system supports the manual modification of the otherwise-equal values. In this way, it is possible to provide higher weights for the most relevant elements. Once some probability values are modified, the system automatically re-adjusts the values to ensure that the sum of all the probabilities is one.

The keyword-based pull recommendation process proposed is presented in Figure 4.3, which summarizes the following sequence of steps:

1. Input of the query: the user introduces the keywords as the input query in the system.

2. Query pre-processing: the keywords are preprocessed by using the extension of

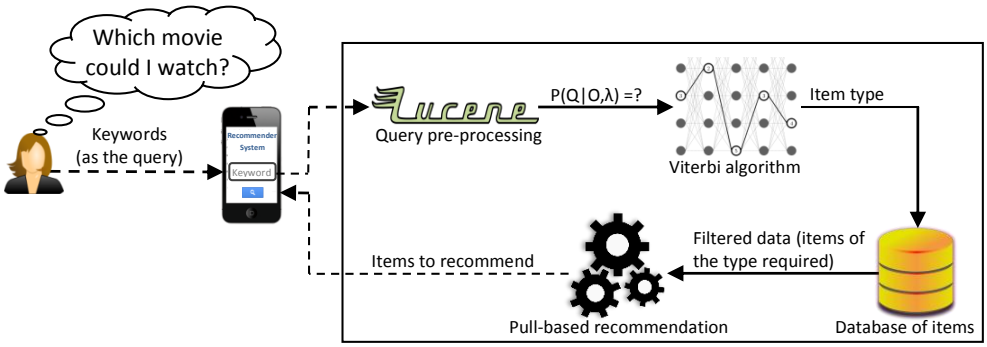


Figure 4.3: Keyword-based pull recommendation process by using HMM.

an analyzer of Lucene 2.4.0 [Apa05], which applies some filters:

- Quotation tokenizer: it parses the query respecting the numbers and the double quotes.
- Standard filter: it applies a standard tokenizer that parses the query based on a grammar (e.g., it splits words at punctuation characters, it removes punctuation, it splits words at hyphens, unless there is a number in the token, and it recognizes email addresses and internet host names as a single token).
- Lower case filter: it normalizes the text of the token by converting it to lower case.
- Stop filter: it removes stop words from the token stream, by using a file with stop words.
- Snowball filter: it applies a filter that stems words using a Snowball-generated stemmer [Por01].

It should be noted that the file "observations.txt" has been previously preprocessed by following this same method.

3. Application of the Viterbi algorithm: given the keywords as the observation sequence O and the HMM model λ, it allows determining the state sequence Q with the highest probability (e.g., music-title, music-artist, music-category, book_isbn, book_title, book_author, book_year, book_publisher, etc.).

4. Selection of the type of item: the item type that the user needs would be determined by the highest-frequency state sequence (obtained in the previous step).

5. Filtering of the database: the database containing the different datasets is filtered by considering the type of item identified in the previous step (e.g., film, music, book, or concert). The data filtered will be used by the pull-based recommendation algorithm.

6. Application of the pull-based recommendation algorithm: it allows obtaining items of interest as an answer to the query submitted by the user, by applying any existing recommendation algorithm desired.

7. Display of the items recommended: a list of items recommended are provided to the user.

For the implementation of the HMM-based method, we used the Hidden Markov Model functionalities provided by the popular library *Apache Mahout* [Apa14].

4.1.6.2 Information Retrieval Approach

A second solution to consider to solve our general problem is the use of IR techniques [SM86, FBY92, BYRN99, MRS08]. In the area of Information Retrieval (IR), where generally the data are unstructured (e.g., searching relevant documents in the Web), the problem of keyword-based query answering by using an inverted index [ZM06] has been studied. For structured data, the field of keyword-based search has started to emerge more recently [CSS10]. There are several systems that support keyword-based searching over structured sources, such as DISCOVER [HP02], KEYMANTIC [BDG+10], KEYRY [BGRV11], BANKS [ABC+02], and DBXplorer [ACD02]. For example, a generic keyword search method was presented in [LOF+08], named EASE, which allows indexing and querying large collections of heterogeneous data (unstructured, semi-structured, and structured data). The authors extended the traditional inverted index in order to provide keyword-based search, as well as proposed a novel ranking mechanism to improve the search effectiveness.

For this case, the index of the retrieval engine contains a certain number of documents, whose content is obtained from the databases automatically. Each document is named with the item type and the feature names. For example, for the dataset InCarMusic [BKL+11], the document names to index are "music_title", "music_artist", and "music_category". The content of each document is composed of the values of the features (e.g., the artist names, the music categories, and the music titles), the item type (e.g., music), and the names of the features (e.g., title, artist, and category). The structure of the documents to index is displayed in Figure 4.4.

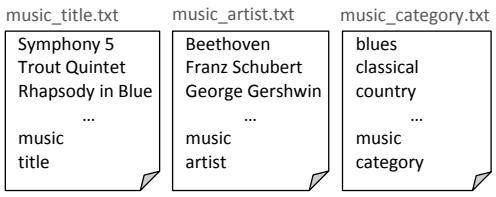


Figure 4.4: Example of the structure of the documents to index with the IR approach.

In general, the keyword-based pull recommendation process performs the following steps (see Figure 4.5):

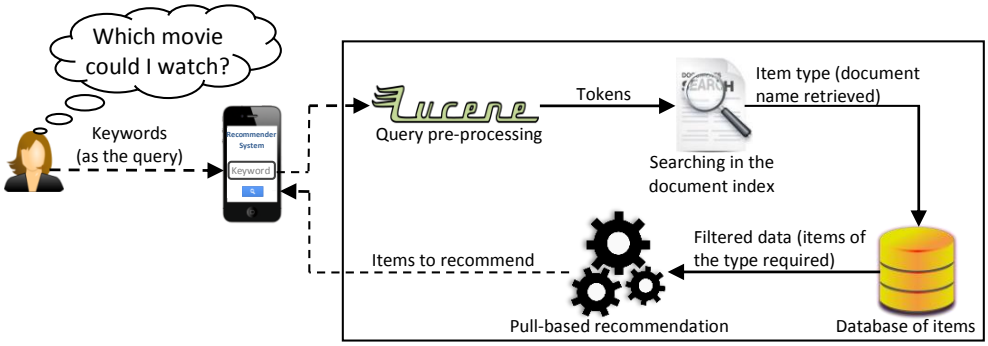


Figure 4.5: Keyword-based pull recommendation process by using IR techniques.

1. Input of the query: the user introduces the keywords as the input query in the system.
2. Query pre-processing: the keywords are preprocessed by using the same procedure described for the HMM approach.
3. Application of the IR algorithm: given the keywords, the system searches in the index the k documents that are the most relevant to the query.
4. Selection of the type of item: the item type that the user needs would be the item type corresponding to the most relevant document (of the ranked list).
5. Filtering of the database: the database containing the different datasets is filtered by considering the type of item identified in the previous step (e.g., film, music, book, or concert). The data filtered will be used by the pull recommendation algorithm.
6. Application of the pull recommendation algorithm: it allows obtaining items of interest as an answer to the query submitted by the user, by applying any existing recommendation algorithm desired.
7. Display of the items recommended: a list of items recommended are provided to the user.

In our prototype, for the indexing of the documents for the IR-based method, we used *Apache Lucene* [Apa05]. Lucene is also used for pre-processing (of input keywords and/or documents) in both methods.

4.2 Push-Based Recommendation Approach

In this section, we present a model that allows different stakeholders and entities to be integrated in a push-based recommendation process and articulates the information exchange among the existing participants involved [HITdCRH15]. This model

particularly focuses on the importance of the environment(s) of the user receiving recommendations and also on the set of potential phenomena happening around her/him.

4.2.1 Contextual Model

First, we present the different elements of our model (see Figure 4.6): contexts, environments, agents, users, events, and activities.

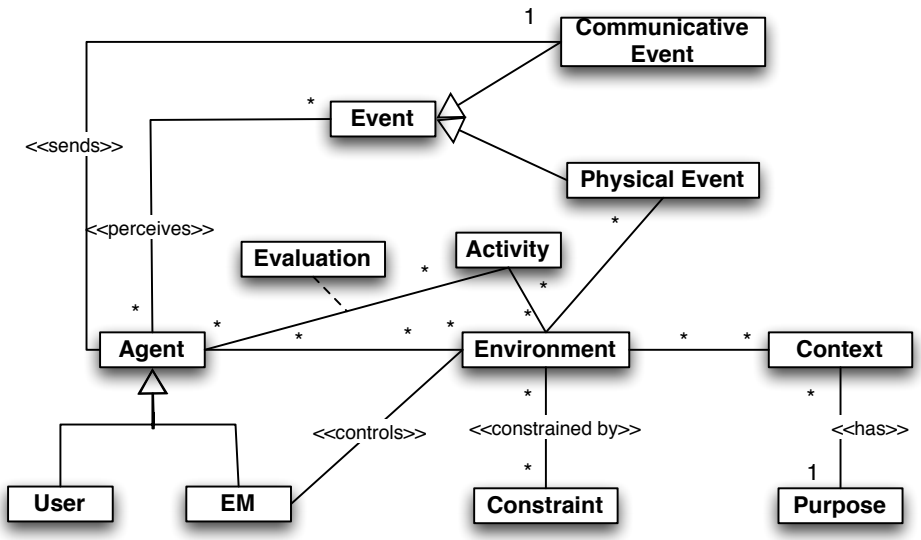


Figure 4.6: Class diagram representing the proposed push-based recommendation model.

Contexts

The main idea behind our model is the division of the users' dynamics into *purpose units*, called *contexts* (Definition 3). A context delimits the scope or purpose of a recommendation. For instance, in the context of *research*, which papers to read next or which top-rated conferences to attend are valid information recommendation outcomes, while suggesting the best route to get to work would be meaningless.

Definition 3 *A context c is a tuple $\langle \mathcal{E}, \delta \rangle$, in which $\mathcal{E} = \{e_1, ..., e_n\}$ is the set of environments in which the user is active and δ is the purpose of the recommendation process.*

It is important to remark that similar purposes with different associated environments may likely result in different recommendations. This is exactly the intention of

our approach: to capture the dynamics in the user's context to produce more accurate recommendations.

Environments

An essential part of our model are the *environments* (Definition 4), since they allow encapsulating a recommendation process and its associated contextual information, as well as the communication among different entities.

Definition 4 *An environment* $e_c = \langle U, \Theta, Act \rangle$*, belongs to context* c*, and is a common area, physical or virtual, in which users* U *(the set of users currently active in the environment) coexist to perform a set of activities* Act *under certain environmental constraints* $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$*.*

The constraints Θ must be understood as physical or virtual boundaries used to unambiguously delimit the environment. Some (but not all) of the characteristics that might be used to form Θ are the following:

- Physical geo-location: coordinates of the physical area that determines the boundaries of the environment (e.g., a quarter or a shopping mall).
- Virtual location: for instance, the web site of a social network the user has just logged in.
- Time: the period of time the environment is valid for (e.g., a *concert* environment will be valid during the duration of that event).
- Number of users: some environments might accept only a limited number of users, for instance when exceeding a certain number of users could entail important difficulties (lack of physical space, risk of overcrowding, communication failures due to bottlenecks, and so on).

It is important to highlight the existing differences between the contextual purpose δ and the environmental constraints in Θ. While the former delimits the topic of the recommendation, the latter constrains the physical and virtual limits for a user to belong to an environment. Moreover, environments do not only act as mere containers for users but they are also meant to be a source of information for them. Thus, besides accommodating the existence of users and enabling communications among them, environments generate (or allow external sources to generate) information relevant within the environment by means of *events* (e.g., messages providing useful information).

Users may be active in several environments at the same time, which implies that environments can overlap. As an example, someone might be visiting a museum but also be part of the environment of the quarter the museum is located in; in that case, she/he could receive relevant information about the museum and about the surrounding area at the same time.

Agents

Agents represent actors in the recommendation process that generate *events* (in particular, *communicative events*) that can affect users in a environment, and are denoted by the set $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. There are two fundamental types of agents: *users* and *Environment Managers (EMs)*. A user is an interested party in recommendations in an environment or an entity that provides related information, while an EM is a special agent associated to a specific environment in charge of controlling the membership of users and communication issues in the environment (as explained later in Section 4.2.2). Users in the environment must be understood either as final users (consumers of the recommendations) or representatives of third parties in the environment (e.g., different types of businesses offering specific services). For example, in the environment of a shopping mall, there might exist some special users representing different shops, announcing (broadcasting messages) special offers for other users nearby (potential customers).

Users

As mentioned before, *users* are a type of agent. A user u_i belongs to an environment e_c if she/he fulfills all the constraints Θ of that environment. Formally, let $e_c = \langle U, \Theta, Act \rangle$ be an environment, then:

$$belong(u_i, e_c) \leftrightarrow \forall_{\theta_j \in \Theta}\ fulfills(u_i, \theta_j)$$

Consequently, the user u_i leaves an environment she/he belonged to iff any of the constraints in Θ is no longer fulfilled by her/him. Then, for example, when a user leaves a shopping mall it makes no sense to keep her/him informed about any information related to that environment.

We consider that users have a *multi-attribute utility function* that guides their decision process (see Equation 4.2). Users are assumed to be rational and perform actions that maximize their *utility functions* (i.e., selecting items and activities that better fit their interests). Given a set of attributes $X = \{x_1, x_2, ..., x_n\}$, the utility is modeled by using a function combining single-utility values obtained by evaluating the attributes of activities in which a user is interested. A vector $\hat{\omega}_{u_i} \in [0,1]^n$ ($<\omega_1, \omega_2, ..., \omega_n>$) defines the relevance of each attribute for the user u_i, and $u_i(x_j)$ is the single utility (i.e., a score) of the attribute x_j for a specific user u_i. Thus, this vector represents which attributes she/he considers more important, and could be defined explicitly by the user according to her/his preferences. The overall utility function is determined as follows:

$$\mathcal{U}_i = \sum^{j} \omega_j u_i(x_j) \tag{4.2}$$

Events

Two types of *events* (Π) are considered: *physical* (Π_p) and *communicative* (Π_c). The former are measured by sensors and represent uncontrollable phenomena that can occur due to the inherent nature of the environment, such as weather conditions, traffic congestions, etc. The perception of *physical events* will be usually carried out by means of *physical sensors* [LML$^+$10], which could collect diverse information such as the GPS location, temperature, light, signal connection strength, etc. Nevertheless, events measured by *virtual sensors* (e.g., the detection of an event in the user calendar, a message posted by the user to a social network, etc.) are also considered physical events. Communicative events released by agents are presented as a way to inform users in the environment about different issues. For instance, tourists in a museum could be informed by an EM associated to the museum about the stop of sales of tickets as a safety measure to avoid crowds in the exhibition halls. Another example of this type of communicative events would be a broadcast advertisement of a new offer for a short period of time in a certain shop in the environment of a shopping mall.

Users are assumed to be able to capture any event addressed to them. Physical events can be detected by any user belonging to the environment. However, *communicative events* might target only a subset of users in the whole set of users U. As an example, a certain offer in a shop might be of interest only for users in a specific age range (e.g., people that are 18-25 years old, who may possess a youth card that entitles them to specific discounts), and so users out of the intended age range could be filtered out. It is important to avoid providing events to the wrong target group, as this could be a nuisance and lead to an eventual desensitization to events and the recommendation system in general (i.e., lost of interest in the recommendations provided, with a potential eventual ignorance of all of them).

Activities

Regarding our proposed push-based recommendation model, we prefer to use the term *activity* instead of the concept of *item*, traditionally used in the field of recommendation systems and used in the rest of this thesis. We believe *activity* is a more appropriate term for push-based recommendation, in the sense that it represents an action on an item rather than simply an item (e.g., watch a movie in a cinema vs. the movie itself). Activities are denoted by the set $Act = \{act_1, ..., act_n\}$ and represent the actions that the user can carry out in an environment. So, they are more specific; for example, a user can be recommended to buy a certain book, to read it, to rent it, etc., depending on the circumstances.

4.2.2 Management of Environments

EMs poll the users with periodic messages indicating which constraints they must fulfill to be active in the environment; we denote these messages with $poll_{EM \rightarrow u_j}(e_i, \Theta_i)$. Once a user u_j receives a poll message, her/his mobile device checks whether she/he

was already active in that environment e_i by querying an *environment table* (ET_{u_j}) stored locally on her/his mobile device.

If the user was not active in that environment, then her/his mobile device proceeds with the verification of constraints regarding the information collected by its sensors. If the constraints are satisfied, then it replies to the EM with an acknowledgement $ACK_{u_j \rightarrow EM}$. When receiving this message, the EM adds (or updates) the state of the user, setting it to *Active*. If the acknowledgement message is never sent from the user u_j, then the EM simply ignores the user until the next poll. This might be caused because of the user's incapability to fulfill the constraints any longer, and so the user is not considered anymore as an active user by the EM. This processing flow is depicted in Algorithms 8, 9 and 10. We consider that a user has also left an environment when a period of time (timeout) has passed since the last message received from the corresponding EM; for simplicity, in the algorithms we consider that this timeout equals the poll period.

Algorithm 8: EM's communication of poll messages

```
1 while (true) do
2     send(u_j, POLL, environmentID, Θ);
3     sleep(secondsBetweenPolls);
```

Algorithm 9: User's message receiving task

```
   Input: The user ID (userID), a map implementing the environment table for
          the user (ET).
1  while (true) do
2      m ← receiveMsg();
3      if (ET.get(m.environmentID) ≠ ACTIVE) && (m.type == POLL))
        then
4          if (checkConstraints(m.Θ) == true) then
5              send(EM, userID, ACK);
6              ET.put(m.environmentID, ACTIVE);
7          else
8              if ((ET.get(m.environmentID) == ACTIVE) &&
                (checkConstraints(m.Θ) == false)) then
9                  ET.put(m.environmentID, INACTIVE);
10             else
11                 dropMsg(m);
```

We illustrate and summarize the proposed model with the example depicted in Figure 4.7. In the figure, we can observe a context (with a certain purpose) containing three different environments. Each of the environments has a set of active

Algorithm 10: EM's message receiving task after sending poll messages

Output: U is the set of users in the environment.

```
1 U' ← ∅;
  // For each user, and subject to a maximum time period for the
  // whole process, perform the following:
2 while (true) do
3 |   m ← receiveMsg();
4 |   U' ← U' ∪ m.userID;
5 U ← U';
6 return U;
```

users. Moreover, each environment will be managed by a corresponding EM. Note that environments may overlap and that the same user might be active in different environments at the same time.

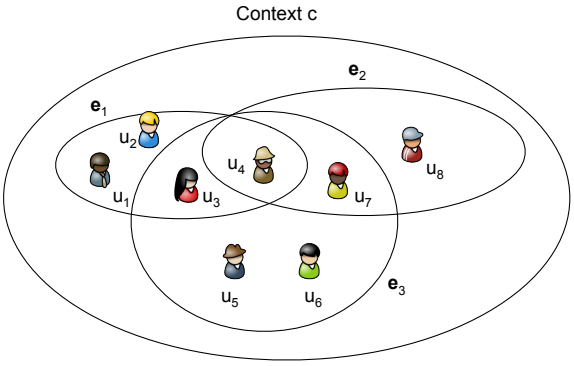


Figure 4.7: An example of a possible contextual model for pushed-based recommendations.

4.2.3 Dynamic Recommendations

In line with the contextual model presented in Section 4.2.1, the recommendation process has to be understood as a *multi-layer process*, in which a user can be active in several environments at the same time. In the model, we have presented environments as overlapped "boxes" containing users, events, activities, and a framework for communication, hosting appropriate RS. The contextual model allows the user's device to obtain recommendations based on the context around. For simplicity, we assume that recommendation algorithms are applied on the nodes where EMs run and that the mobile devices of the users apply local post-filtering and prioritization

based on preferences and private data stored only on the mobile devices.

Recommendations are encapsulated by a function as follows [ASST05]: $f_r : U \times Act \times C \rightarrow R$, where U stands for a set of users, Act is the set of activities to be recommended, C is the context for the recommendation, and R are the recommendations provided (usually presented as a ranking of activities).

A push-based recommendation process implies that the user would not need to explicitly request recommendations about specific types of activities she/he is interested in; instead, the system will automatically create recommendations and provide relevant activities to the users proactively, even in the absence of user requests. Thus, the RS would communicate the results of significant recommended activities once they had been already assessed. In order to achieve that, several key aspects must be addressed:

- A *recommendation triggering* approach to decide when the recommendation process should start.
- A *pre-filtering phase* to filter out activities out of the scope of the user's context.
- A *recommendation algorithm* to use for the remaining set of activities.
- A *post-filtering phase* for conflict resolution between activities (recommendations for different overlapped environments).
- A way to present the results to the user (*results display*).

Figure 4.8 summarizes these phases, indicating the proposed workflow and whether they are executed in the user's device or by the corresponding EM.

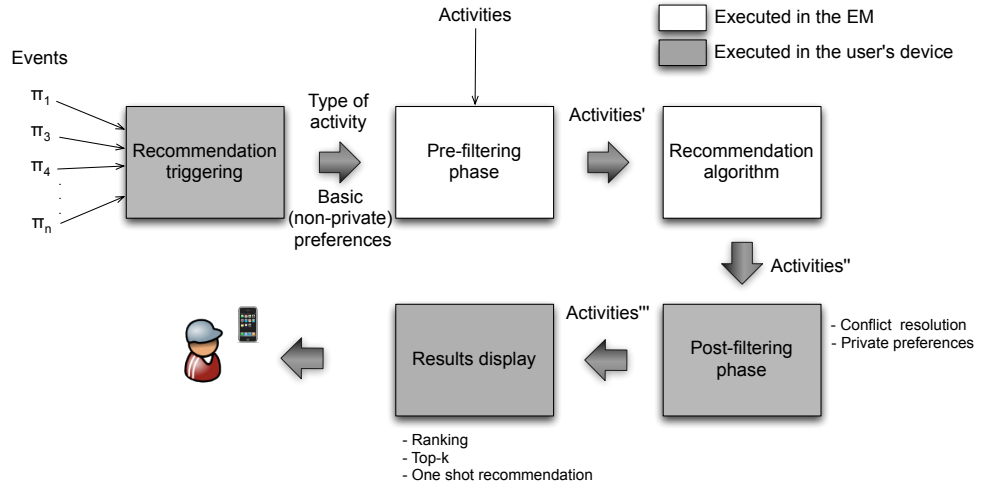


Figure 4.8: Recommendation process workflow for the push-based approach.

Notice that the properties describing the activities offered in the environments (i.e., the *activity properties*) and the attributes of the user's *utility function* are in general different sets. Let $\beta_{Act_i} = \{\beta_1, \beta_2, ..., \beta_m\}$ be a set of properties describing an activity and $X_u = \{x_1, x_2, ..., x_n\}$ be the set of attributes that affect the assessment of the user's utility of the activity. Then $\hat{p}_{Act_i} = <p_1, p_2, ..., p_m>$ is defined as a property vector for activity Act_i, where p_j represents a rating according to, at least, a user's opinion about the activity property β_j. Thus, for example, taking Act_i as *dinner at Chez Léon* and $\beta_{Act_i} = \{$location-distance, avg-price$\}$, an example of a property-value vector is $\hat{p}_{Act_i} = (2.3\ km, 30\ €)$.

With the above description of activities, we need to redefine the utility function of users, in order to include the specific activity property-values available into the assessment. The reason is that attributes in X_u can be generic (depending on the user), such as the preference for quiet areas, distaste for luxury places, and so on, and these attributes may not match perfectly with properties describing activities. For instance, in TripAdvisor [KS+00] some of the properties for a ranked hotel are its *location*, the *service* provided, or the *cleanness*, but nothing is explicitly rated on the serenity or friendliness of the area where it is located. For the matching process between the attributes in the user's utility function and the specific activity properties, a semantic alignment of the user's attributes affecting her/his utility and the properties describing the activities to be recommended could be performed. We let the problem of concept alignment between the user's attributes and the activity properties to the implementation of the function $u(\cdot)$ used in Equation 4.3 (see below): an activity property $\hat{p}_{Act_k}$ is matched to an attribute x_j in the user's utility function considering a strength ω_j (relative weight regarding the contribution of their matched utility to the global utility).

$$U_i^{Act_k} = \sum^{j} \omega_j u(x_j, \hat{p}_{Act_k}) \tag{4.3}$$

Recommendation Triggering

In a push-based recommendation, some conditions must be detected to trigger the recommendation process without user interaction. These conditions will be defined by means of different events or circumstances taking place in the environments where the user is active in, such as weather conditions, the user's geo-location, the messages that a user can receive (regardless of the channel used), such as special offers in a mall or messages from other users indicating that something is happening, etc.

A function $t : \mathcal{P}(\Pi) \times TypeActivity \rightarrow \{true, false\}$ is defined as the triggering function, where $\mathcal{P}(\Pi)$ is the power set of events perceived in the environment and $TypeActivity$ is the set of the types of activity the user may be interested in, for instance, *have dinner*; the types of activities relevant in an environment have been previously sent from the corresponding EM to the device. Different mechanisms could be applied to define appropriate triggering functions for specific users and types of activities/items, based on the preferences of the users.

It should be noted that the recommendation triggering phase is executed on the

mobile device of the user (see Figure 4.8), even though we are considering push-based recommendations. Although the triggering could also be performed by the EMs, this would require the constant communication of data to the EM by the mobile device (location, context data, user preferences, etc.), which would be costly and also potentially subject to privacy concerns. Anyway, the key aspect is that appropriate recommendations will be created and provided to the user without an explicit user request. Therefore, the overall goal of the recommendation triggering phase is to detect a type of activity that may be relevant for the user (if any), thus triggering the whole recommendation process.

Notice that this proactive recommendation behavior implies a double decision: decide if it is appropriate to trigger the recommendation process and decide which type/s of activities is/are relevant. For this, the purpose of the context of the user is considered.

Pre-filtering Phase

The recommendation process needs to carry out a pre-filtering process to select, among all the instances of the type of activity resulting from the triggering, those that are more relevant in terms of the user's utility function maximization. This will be carried out by the EMs of the environments where the user is active in. To enable this task, we assume that the user's device is able to send relevant information to the EMs. Then, the EMs compare activities' properties and user's preferences in terms of attributes of her/his utility function.

Mathematically, the pre-filtering phase is defined as a function (see Equation 4.4), where Act' is a set of refined instances of a type of activity and $\hat{\omega}_u$ represents the relevance of each attribute for the user. The EM carries out the task of assessing the utility of every instance of that type of activity on behalf of the user. Thus, the user's device must send $\hat{\omega}_u$ and X_u to the corresponding EMs of the environments involved in the context. Every EM will return the activities that better match the preferences of the user, in terms of the corresponding weights and attributes, according to the properties describing the activity.

$$f_{pf} : Act \times \hat{\omega}_u \times X_u \times \beta_{Act} \rightarrow Act' \tag{4.4}$$

Note that this is not an essential phase, but its use is interesting to reduce the amount of candidate activities to be considered as input for the recommendation algorithm.

Recommendation Algorithm

Different techniques could be applied to perform the recommendation process, such as collaborative filtering, content-based recommendation, or a hybrid approach. The input of any of these is a set of instances of the types of activities in which the user may be interested. The output is a set of activities, not only relevant from the user's point of view but also taking into account related content (content-based filtering)

or/and the similarity among users or items' ratings in the environment (collaborative filtering).

Post-filtering Phase

As commented before, in the pre-filtering phase the user's device has to exchange some information with the EM, but this information might lack some details which could be eventually important to provide an accurate recommendation, as it is reasonable to think that the user does not want to share some sensitive information, due to privacy concerns. So, in a post-filtering phase, the user's device exploits this private information to filter out activities obtained by the recommendation algorithm.

As we pointed out, environments may overlap. For instance, in a museum we could have recommendations to visit specific areas of the museum and also recommendations for other activities in the surrounding area where the museum is located. So, a conflict among recommendations for activities in different environments may arise. Different techniques can be applied for conflict resolution (e.g., [ABC+05, RC03]), such as, for instance, priority rules.

Display of Results

Different policies can be considered when displaying the outcomes of the recommendation process on the user's device. A typical approach is to sort the activities by means of a ranking, based on the degree of suitability for the user (calculated in previous phases, especially through the recommendation algorithm), and show the top-k activities. Nevertheless, any other type of ranking would be allowed as well. Furthermore, there exists the possibility that the user prefers not to be annoyed with a number of different possible activities, trusting the RS in such a way that she/he prefers to see only one activity as the output of the recommendation process (i.e., the best activity that the system can suggest).

4.2.3.1 Case Study: A Tourism Scenario

In this section, we present an example tourism scenario in which a user, called Alice, benefits from the multi-layer approach proposed (see Figure 4.9). Alice prefers to avoid crowded places, she usually has lunch around 1PM, and she is allergic to gluten. Now, she is enjoying holidays in Paris for a week. It is her first time in Paris, she loves art, it is raining, and it is almost noon. She initially decides to walk around the Louvre Museum area. Alice's mobile device could easily infer the purpose for the recommendation context as *leisure time*, since for instance Alice could have previously booked those days as vacation time in her calendar.

Management of Environments. The recommendation context is formed by all the environments where Alice is active, and the purpose of the recommendation would be *leisure time*. One environment affecting the user is the museum itself. Besides, the area where the Louvre museum is located (i.e., the Louvre Quarter) is also covered by

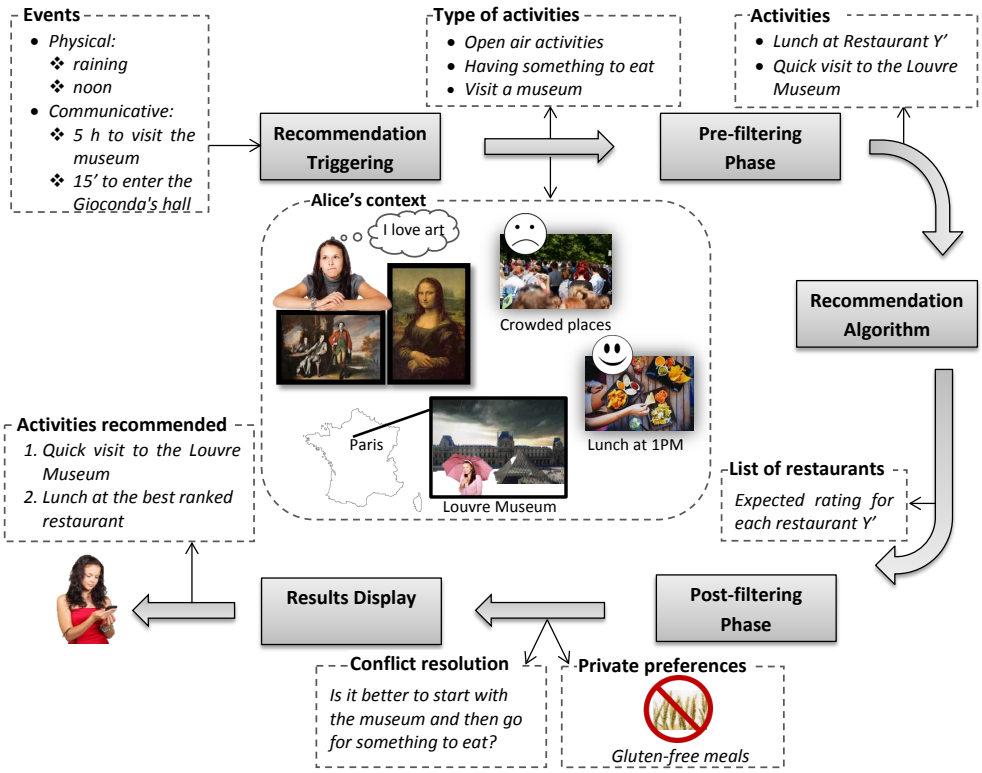


Figure 4.9: Overview of a case study for push-based recommendations.

another environment, which offers the following types of activities: visits to museums in the area, having something to eat, and open air activities (such as running a marathon taking place that very same day). The only constraint (in Θ) belonging to this last environment is to be located within the GPS coordinates of the corresponding area. However, the Louvre Museum has other constraints to accept visitors as active members of its environment: being located within the spatial area of the museum and having activated a QR code provided in each of the entrances of the building. Alice's mobile device and the EMs corresponding to the Louvre Quarter (EM_Q) and the Louvre Museum (EM_M) are assumed to exchange messages as described in Section 4.2.2.

Recommendation Triggering. Alice's device receives events from different sensors and the two EMs involved in the recommendation. It is still raining (physical event); an agent (a special user) representing the museum informs that the average estimated time to visit the museum is around five hours (communicative event) and that the estimated time to enter the Gioconda's hall is about 15 minutes (communicative event); moreover, it is noon (physical event). Alice is willing to enjoy the day, and

she is within appropriate environments for leisure activities, so the recommendation process is triggered. Given the current time of the day, *open air activities*, *having something to eat* and *visit a museum* are considered as feasible types of activity. So, the mobile device communicates to each EM its interest in the instances of these types of activities.

Pre-filtering Phase. When Alice is at the entrance of the Louvre Museum thinking what to do, Alice's device sends the $\hat{\omega}_{Alice}$ and X_{Alice} attributes to each reachable EM to filter out the different instances of the types of activities selected. Let the set of instances available for the potential types of activities be: *lunch at Restaurant* Y (where Y includes all open restaurants in the area), *Regular visit to the Louvre Museum*, *Quick visit to the Louvre Museum*, and *Detailed visit to the Louvre Museum*.

In the case of restaurants, all of them are located outside the museum, and they are at a reasonable distance by foot from the current location of Alice. The EM_Q checks in different websites the set of properties and opinions of former customers. Some of them do not seem to be quiet places, so they are filtered out based on Alice's preferences. On the other hand, in the information exchanged between the device and the EM_M, it is clear that Alice would not like to spend five hours visiting the museum, since that would delay her usual lunch time. Moreover, she does not like crowded places, so maybe she would like to skip some halls with too many people. The EM_M decides to offer a quick visit to the museum. This would allow Alice to visit some master pieces as well as finish at a reasonable time to go for lunch. Then, the result of the pre-filtering phase is a set of the following activities: *quick visit to the Louvre Museum* and *Lunch at Restaurant Y'*, where Y' represents the set of restaurants still interesting for the user.

Recommendation Algorithm. At this point, as the EM_M has only one possible recommendation activity to evaluate (*quick visit to the Louvre Museum*), we focus on the EM_Q, which will for example use information of other users to evaluate the list of restaurants (user-user collaborative filtering). Thus, it could calculate the similarity of Alice with all other participants in the environment who already ate in those restaurants. Next, based on these similarities, it could calculate an expected rating for each restaurant.

Post-filtering Phase. In this phase, a couple of issues must be checked. Firstly, the device must observe if there exist recommendations from different environments. In this case, Alice has several options: recommendations to have lunch (from the EM_Q) and a suggestion for a quick tour in the Louvre (provided by the EM_M). These conflicts among recommendations must be solved. As Alice is used to have lunch a bit later, the device decides that it is better to start with the museum and then go for something to eat. Moreover, due to the fact that Alice does not want to share her celiac condition (allergy to gluten) with any EMs, her own device would filter out those restaurants not offering gluten-free meals (based on content features about

the type of food they serve), even if they were highly rated by the recommendation algorithm.

Display of Results. Finally, a sorted list with the two recommendations would be displayed on the screen of Alice's device, giving Alice a short-time plan of what to do next in Paris: the quick tour to the Louvre and the lunch at the best ranked restaurant.

4.3 Example of a Trajectory-Based Recommendation Approach for Mobile Users

In this section, we present an example of a user-based collaborative filtering recommendation approach for mobile users that considers context data, such as the location of the user and her/his trajectory, to proactively push new up-to-date recommendations to the user in real-time. It could be integrated into the pull-based recommendation approach (see step 3 "Creating/updating a 2D recommender" in Figure 3.4) and push-based recommendation approach (step 3.2 in Figure 3.5), described in Sections 4.1 and 4.2, respectively.

Figure 4.10 shows the workflow of the proposed trajectory-based recommendation approach, which consists of the following steps:

1. Through *user-based collaborative filtering* (*UBCF*), other users with similar preferences to the user are found.

2. The known ratings provided by those similar users are considered to estimate the potential ratings that the user could provide for different items.

3. In order to determine the top-k items to recommend (i.e., the k items with the best predicted ratings and not yet seen by the user); if the list of candidate items is empty, due to the absence of enough data for the UBCF to provide results (cold start problem), then just the k nearest points of interest are collected as candidate items (this is called the nearest POI, or NPOI strategy).

4. The items with a rating prediction above a *recommendation threshold* (e.g., 2.5, given an evaluation scale from one to five) are added to the list of potential recommendations.

5. The resulting list is re-ordered, if necessary, in order to minimize the distance that the user will need to traverse to access those items. For that purpose, the shortest path passing through all those k items is computed and that path is the one recommended to the user.

Moreover, as illustrated at the bottom of Figure 4.10, there are several circumstances that could trigger a reevaluation of the recommendation process:

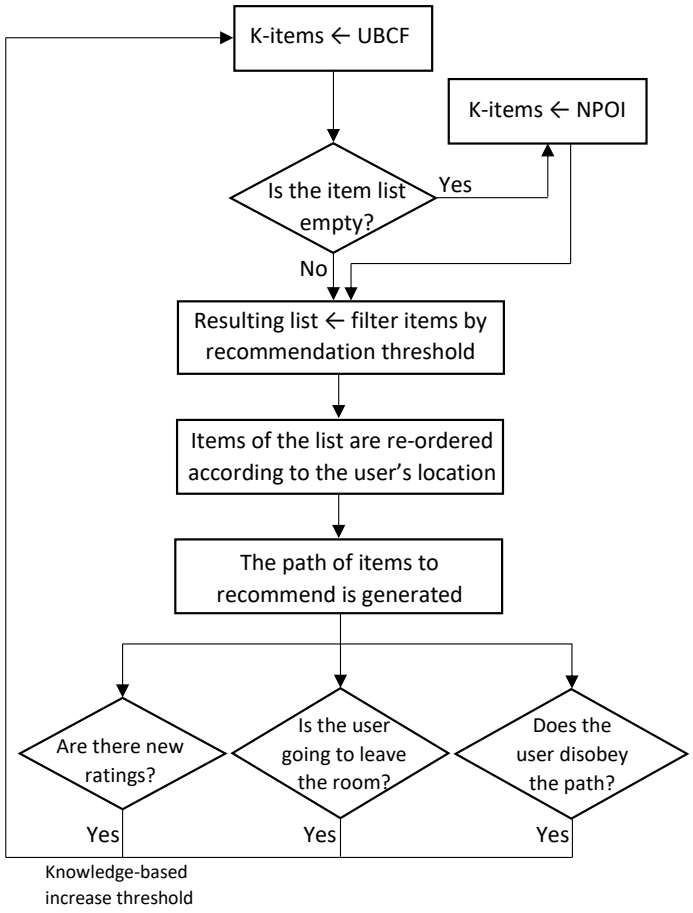


Figure 4.10: Recommendation process for a trajectory-based recommendation approach for mobile users.

- When the recommendation system has significant new information (it has new data regarding at least a certain number of ratings, provided by the user himself/herself or by other users, according to the required *knowledge base increase threshold*).
- When the user is about to leave an area such as a room in a building (and therefore it is convenient to check if there is any additional item in that area worth visiting at that moment).
- When the user has deviated from the recommended trajectory significantly (e.g., because something not recommended attracted her/his attention).

Besides, in order to avoid recommendation instability, a *minimum time interval be-*

tween successive recommendation updates is considered. Finally, independently of the previous conditions, the recommended list of items is updated if the list becomes empty because the user already observed all the items previously recommended. The previous rules ensure that an appropriate list of recommendations can be automatically maintained up-to-date in a suitable way.

4.4 Summary of the Chapter

In this chapter, we explained the two main modules of the architecture proposed in this thesis. First, we described in detail the design of the pull-based recommendation module, which accommodates the pre-filtering, post-filtering, and contextual modeling paradigms. This module provides reactive recommendations, obtained as answer to a query explicitly submitted by the user and evaluated by the system as a continuous query. We also introduced a similarity metric to compare contexts in the pre-filtering and post-filtering paradigms. In the contextual modeling (used in pull-based recommendations), we consider the recommendation model as a classification problem, where the context variables are the features and the ratings are the decision classes; in this way, we do not need to use a similarity metric because we include the contextual information in the recommendation function as part of the estimation of ratings. The pull-based recommendation module assumes by default that the user enters into the system the item type of interest (e.g., restaurant, movie, book, museum, etc.). Hence, we described two possible strategies to the problem of identification of the item type from keywords specified by the user in a pull-based recommendation: an alternative proposed is based on the use of the Hidden Markov Model and the other one exploits Information Retrieval techniques. Finally, we presented a generic push-based recommendation model for mobile users. The proposed model is a multi-layer model, it is general, and it can be adapted to different mobile computing scenarios and domains. Besides, it is based on a solid definition of the concepts of context and environments, takes into account the impact of dynamic events, and includes all the actors that may play a role in a mobile context-aware recommendation process.

Chapter 5

DataGenCARS

The capability to generate context information is a key feature of DataGenCARS, as existing real datasets often lack rich context data, which implies that it is very difficult to exploit those datasets to evaluate context-aware recommendation algorithms. For example, as we analyze in detail in Section 8.2.1, the STS dataset [BERS13, EBRT13] contains 2534 ratings (on a scale from one to five), provided by 325 users in different contexts, of approximately 249 POIs, and using 14 context dimensions to characterize the context of the user. Unfortunately, our detailed analysis of that dataset shows that it is very sparse in terms of the availability of ratings and contexts; for example, the context variable "transport way" is specified only for 3.2% of the ratings and the most-frequently specified context variable (the "temperature") appears only in 15.6% of the ratings. As another example, only a maximum of five context variables are provided for some ratings and most ratings have only one context variable defined (or none). Moreover, for 23.64% of the ratings in the dataset only one context variable has a defined value and 38.44% of the ratings have none.

In this chapter, we present DataGenCARS, a complete Java-based synthetic dataset generator that can be used to obtain the required datasets for any type of scenario desired, allowing a high flexibility in the obtention of appropriate data that can be used to evaluate context-aware recommendation systems. In Section 5.1, we describe the main functionalities of the generator. Then, we explain the most important classes of the proposed architecture. In Section 5.2, we illustrate the use of DataGenCARS to perform some interesting tasks that can be useful to generate data for the evaluation of recommendation algorithms. DataGenCARS presents features such as: a flexible definition of user schemas, user profiles, types of items, and types of contexts; a realistic generation of ratings and attributes of items; the possibility to apply different workflows, such as the generation of a synthetic dataset similar to an existing one, a completely-synthetic dataset, and a dataset of ratings incrementally; the completion of missing information in datasets; and the composition of workflows. In Section 5.3, we present a summary of the basic tasks that can be performed to evaluate a synthetic dataset generated with DataGenCARS.

5.1 Basics of DataGenCARS

In this section, we first describe the functionality of DataGenCARS [dCRHIHTL17a]. Then, we present an overview of the main classes of its architecture. The input files and the output data files it produces are described in Appendix B.1.

5.1.1 Functionality of DataGenCARS

DataGenCARS is a tool that supports the generation of synthetic datasets of users, items, contexts, and ratings, which can be used to evaluate CARS. Figure 5.1 describes the general functionality of the dataset generator. Note that the tool may need different phases depending on the purpose of the dataset to be generated, namely: i) create a new dataset from the scratch; ii) create a new dataset replicating the features of an existing one; or iii) enlarge an existing dataset. Some key features of DataGenCARS are described in the following.

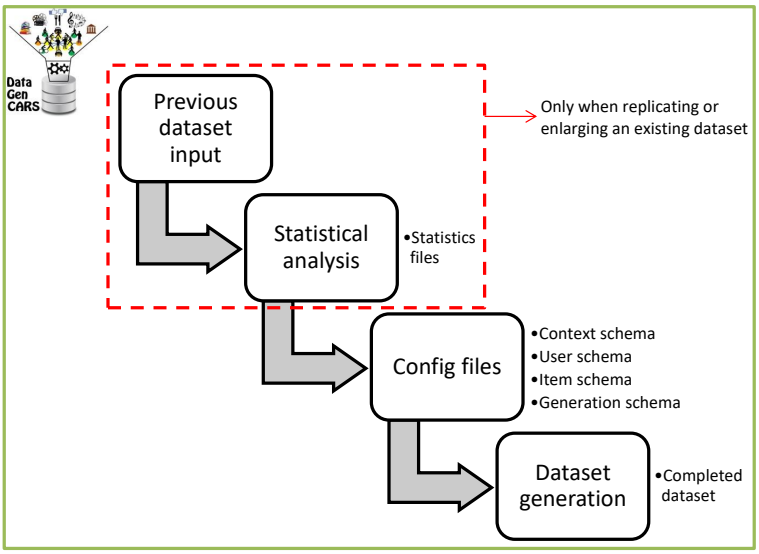


Figure 5.1: Simplified workflow of DataGenCARS.

Flexible definition of user schemas

A *user schema* is the set of *attributes* or *variables* that characterize a user, including the names of the attributes, their domains, and possible additional constraints (e.g., uniqueness when the value of the attribute cannot appear more than once in the set of users synthetically generated). DataGenCARS supports the definition of any desired user schema. For example, it might contain attributes such as the age, gender, occu-

pation, etc. The values of these attributes could be useful to evaluate recommendation algorithms that exploit features of the users to perform the recommendations.

Flexible definition of context schemas

A *context schema* defines the features of a context. DataGenCARS supports any desired definition of the attributes representing the users' contexts. For example, we might want to define the context through the set of attributes "day of the week", "mood", "temperature", "location", and "type of social situation".

Flexible definition of type of item schemas

A *type of item schema* defines the features of a type of item. DataGenCARS supports the desired definition of the attributes for the synthetic items generated. Thus, we could define the type of item "restaurant" with attributes "address", "telephone", "price", "opening hours", "type of food", etc. DataGenCARS can generate a dataset with any desired number of items for each of the item types defined.

Flexible definition of user profiles

Besides the user schemas, it is possible to set up different user preferences for items by defining different data *utility functions*, that we call *user profiles*, and assigning them to the generated users. A user profile is defined by means of a list of attributes (attributes of the type of item schema and/or context schema) and the weights to apply to the values (*scores*) of those attributes to compute an overall score that represents the user's utility of an item in a specific context. The sum of all the weights in a user profile must equal one; nevertheless, DataGenCARS provides an automatic readjustment of the weights (respecting their relative values) to make sure that summing them all gives one. The weights in the description of a user profile may have the associated symbols (+) or (−): the former indicates that the order of preference of the values of the attribute for that user profile starts with the furthest on the right in the sorted list of possible values, while the latter indicates the opposite. Non-relevant attributes defined in the scheme file have a weight of 0.

Moreover, the user profile can include an attribute *others* representing unknown factors or noise: this allows modeling realistic situations in which the user profiles are not completely defined and where the rating behavior of the user is partially driven by unknown factors. As an example, the user profile 1 in Figure B.4 (see Appendix B.1.2) considers that a 20% of a rating provided by a user characterized by that user profile is due to unknown factors (notice that a weight of 0.2 has been assigned to the generic attribute *others*). As another example, a user profile with weight 1 for the attribute *others* (and 0 for the remaining weights) would represent users that behave in a completely-unpredictable manner, as the ratings that they provide cannot be explained by any of the attributes defining the user profile.

Realistic generation of ratings

To generate synthetic ratings that could have been provided by real users in a given context, DataGenCARS can consider:

- *The context uncertainty.* We do not assume that users will provide all the information about their context in a precise way. Instead, a configuration parameter allows selecting the desired average context information provided in the ratings. For example, if this parameter is set to 10%, then for each rating only an average of 10% of the context information will be available along with the rating. Needless to say, the specific context attributes filled for different ratings of the users might differ.

- *The user's expectations.* In real life, a user that recently voted bad items and who then sees a good item will probably have a tendency to over-score the new item, and vice versa. With DataGenCARS, it is possible to configure the estimated impact of expectations on the user ratings by configuring the number of recent ratings k to consider and the way that the corresponding ratings could affect a new rating provided by the user.

- *The rating distributions.* It is possible to simulate different distributions of ratings among the users, such as uniform distributions (i.e., users tend to contribute with a similar number of ratings), Gaussian distributions with a certain average number of ratings per user, biased distributions where there are highly-participative users and others that hardly provide ratings, etc.

Moreover, it is possible to simulate changes over time in the parameters defining the three aspects mentioned above (the uncertainty regarding the context, the user's expectations, and the rating distributions), thus simulating the potential temporality of those features (e.g., the rating distributions could change after a certain amount of time, as some highly-participative users could eventually lose interest and stop providing ratings). For this purpose, we could simply generate several datasets for different temporal periods (each generated using different parameter values, as required) and then merging them together.

Notice that any desired value can be set for those parameters, thus supporting the definition of any situation. Therefore, there is no compulsory need of estimating values for the parameters. If reflecting values that are appropriate in a specific real situation is required, designing and developing a social experiment (e.g., by using surveys) could be considered, to try to determine appropriate values for that case. The power of DataGenCARS precisely relies on allowing the evaluation of recommendation approaches in any hypothetical situation required; for example, we can easily check how a recommendation algorithm would behave if little context data is available, if the distribution of ratings among users is highly non-uniform, or if users tend to over-rate items, just to cite a few examples.

Consistent generation of item attributes values

When an item is generated synthetically, two strategies are applied to ensure a suitable consistency among the values generated for the different attributes of the item:

- Each item to generate is previously assigned by DataGenCARS a certain category, which we call *item profile* (e.g., we could represent a static classification of items in generally-objective categories such as "good", "normal", or "bad"). Item profiles support the definition of typical ranges for different attributes whose values are usually correlated. The values of some attributes of the item (the *attributes considered relevant for the item profile*) are generated in a way that is acceptable for the corresponding item profile. So, item profiles are simply a useful tool that can guide the synthetic data generation process in the required way. For example, for a restaurant with profile "good", the value of the attribute "food quality" may be expected to be "excellent" or "good", but it is not very likely to take a value "horrible", as in that case probably the restaurant would not be globally considered as "good". Nevertheless, it is possible to simulate also unexpected values (*consistency noise*, which represents the percentage of variation allowed over the standard range that defines each item profile). For example, in the case of a good restaurant it might be possible to generate a value of "normal" for the "food quality" or even a value of "bad" for the "quality of service", as other attribute values could compensate these disadvantages and still lead to an overall good profile. A consistency noise of $x\%$ represents a potential deviation of up to $x\%$ to the left or to the right of the corresponding range. The required item profiles can be defined in a flexible way and their use can be combined with the definition of appropriate values for the consistency noise. Moreover, if item profiles are not required, we could simply define a single item profile to represent any possible item; so, item profiles offer support for the potential need of correlations among different attribute values for a single item, but the definition of item profiles is optional.

  It should be noted that the concepts of item profiles and consistency noise could be reframed based on the theory of *fuzzy logic* [YL98]. As an example, let us suppose that DataGenCARS needs to generate a "normal" restaurant. The function "food quality for a normal restaurant" could be defined as shown in Figure 5.2. According to that function, a score between 3 and 4 is typical for a normal restaurant, but with a lower probability (lower membership degree) it is also possible to obtain a value of food quality as low as 2 (more typical of bad restaurants) or as high as 5 (more typical of excellent restaurants), with a membership degree varying in this case according to the shape of a trapezoid, which is one of the types of membership functions typically used in fuzzy logic. In this case, the actual value of the food quality score for the normal restaurant generated can be obtained by generating a random value using the trapezoidal distribution [AD72, Wal96, Het12] shown in the figure. Of course, other membership functions (e.g., a Gaussian function) could be used similarly. At this point, the relation with the concept of consistency noise explained above should

be noted. In fact, in Figure 5.2 there is actually a consistency noise of 20% over the attribute's range (assuming that the typical range of the score for a normal restaurant is 3-4, the score could be 20% lower than 3 or 20% higher than 4). The difference with the approach implemented by DataGenCARS is that, for simplicity, in the basic architecture of DataGenCARS the representation of that potential lack of consistency is not gradual but sharper (as shown in Figure 5.3, a rectangle is used rather than a trapezoid). Nevertheless, it is easy to incorporate finer-grained definitions of item profiles using the fuzzy logic approach presented here. The trade-off is that more work is required to define the membership functions for each attribute and item profile; simply defining a consistency noise percentage is easier for the user of DataGenCARS, even though it allows for a lower degree of parametrization.

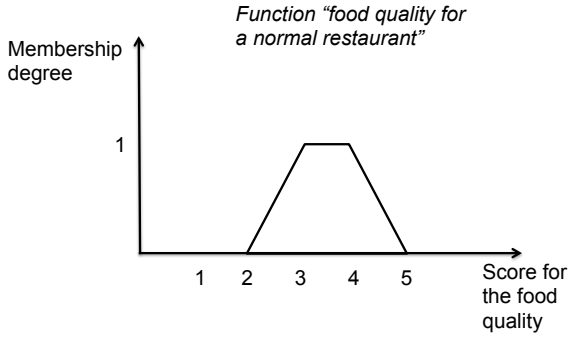


Figure 5.2: Definition of item profiles and consistency noise using fuzzy logic and a trapezoid membership function.

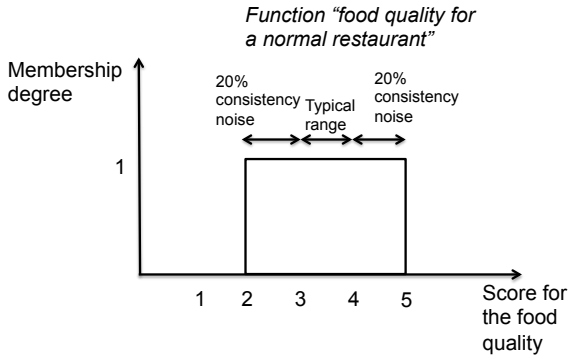


Figure 5.3: Simple definition of item profiles and consistency noise explained using fuzzy logic.

- Besides supporting the definition of item profiles and the attributes relevant for

each item profile, DataGenCARS also allows the definition of *jointly-generated attributes* (also called *composite attributes*, for simplicity), which are sets of attributes that cannot be generated independently due to the existence of strong correlations and interdependencies among them. For example, the attributes "street", "number", "ZIP code", "city", "state", "latitude", and "longitude" can be generated at the same time by querying external geographic data sources (e.g., Google Maps, OpenStreetMap, etc.), to ensure that the resulting address is correct.

The different item profiles and the attributes relevant for each item profile, the consistency noise, and the average percentage of items to generate for each item profile are configuration parameters of DataGenCARS. The jointly-generated attributes are specified in the schemas defining the types of items.

Generation of mixed real-synthetic datasets

DataGenCARS can be used to generate any desired dataset adapted to the evaluation needs, including creating a dataset similar to an existing one (applying similar random probability distribution functions to generate attribute values) or enlarging an existing dataset in a flexible way. We might use it to increase the number of users, ratings, items, and/or the types of items in the dataset. Moreover, we could also use DataGenCARS to complete the amount of context that was provided by the users when rating the items (i.e., to complete the amount of context information characterizing the real ratings). Given an existing real dataset of ratings and/or a previously-generated synthetic dataset, DataGenCARS can also be used to recompute the ratings according to other specific user profiles desired. Finally, we could re-use only some existing elements of a given dataset, like information about the items and their features (e.g., names of restaurants, addresses, etc.), in order to generate a synthetic dataset inspired by real values and/or based on the combination of third-party information. This includes the exploitation of external data sources such as OpenStreetMap [Coa04] to obtain information about real types of items (e.g., business) and their locations.

Support for the automatic mapping between item schemas and Java classes

It is possible to define a Java class to represent a type of item and let DataGenCARS automatically generate the corresponding configuration file representing the schema of that item. Conversely, if we have already defined the schema of a type of item, the template of a Java class (attributes and methods) that represents the corresponding type of item can be automatically generated; in some cases, some special annotations (e.g., references to Java packages that should be imported) may need to be included in the input files describing the schema of the items to customize this conversion. In this way, DataGenCARS facilitates the integration of Java code with the generation of datasets, enabling the transparent integration between the application class model

required and the definition of the files needed to configure the generation of datasets with DataGenCARS. Moreover, some users may prefer defining Java classes representing the types of items rather than defining the schemas of the types of items in the corresponding schema file.

Automatic learning of user profiles from an existing dataset

It is possible to infer appropriate utility functions from an existing dataset (see Section 5.2.1). Those utility functions could then be used, for example, as user profiles exploited to generate additional ratings. To generate the user profiles, DataGenCARS makes use of a class called *GenerateUserProfile*, which uses the LSMR iterative method [FS11], implemented in the API of Apache Mahout [Apa14], to try to learn an appropriate utility function, or user profile, for each user (for more details about user profiles, see Appendix B.1.2). Specifically, the LSMR method solves (for each user) a system of linear equations $A * X = B$, and applies the least-squares method if needed. In our case, A is a rectangular matrix of dimension MxN, X is the vector of weights of the utility function that we want to determine (i.e., the unknowns of the system of linear equations), and B is the vector containing the ratings provided by the user. Each row of the matrix A represents the scores (on a scale of one to five) of the different attributes characterizing the corresponding rating in the vector B, M is the number of ratings per user (i.e., the length of the vector B), and N is the number of attributes characterizing the rating (including item attributes and context attributes). The LSMR method is able to handle situations where $M > N$ (i.e., situations where there are less unknowns than equations, and therefore an exact solution is unlikely) as well as situations where $M \leq N$ (i.e., situations where there may be more unknowns than equations, and therefore there may be no unique solution). In any case, the method tries to minimize the error obtained when applying the utility functions learnt to obtain the ratings. Our current prototype learns a different user profile specific for each user. Nevertheless, the alternative of generating more generic user profiles that could be shared by several users could be analyzed.

5.1.2 Architecture of DataGenCARS: Main Classes

As commented before, the dataset generator has been implemented in Java, following an object-oriented design approach. From the high-level overview of the classes in the architecture, shown in Figure B.1 (see Appendix B.3), we can highlight some classes, explained in the following.

AttributeGenerator. The specification of each attribute (in the different input files defining the schema of items, users, and contexts) includes a reference to a Java class generator that will be in charge of generating values for such an attribute.

As depicted in Figure 5.4, different types of generators can be used. Some predefined generators are provided with the basic architecture of DataGenCARS; for example, *DateAttributeGenerator* allows generating random dates within a certain range of dates required, *AddressAttributeGenerator* generates consistent values for typical attributes representing an address ("street", "number", "ZIP code", "latitude", and

"longitude") by collecting these values from an input file provided, *BooleanArrayListAttributeGenerator* generates an array of Boolean values representing the presence or absence of a certain feature or *Component* (e.g., it fills with true/false the opening days of a business –Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday– or the types of foods served in a restaurant –Italian, Mexican, etc.–, based on the average percentage of true values desired), and *RandomAttributeGenerator* generates a random value (an integer in a given range, a value from an enumerated list, or a Boolean, depending on the domain of the specific attribute). Random values are by default generated according to a uniform probabilistic distribution, but it is possible to parametrize or extend DataGenCARS to use other desired distributions.

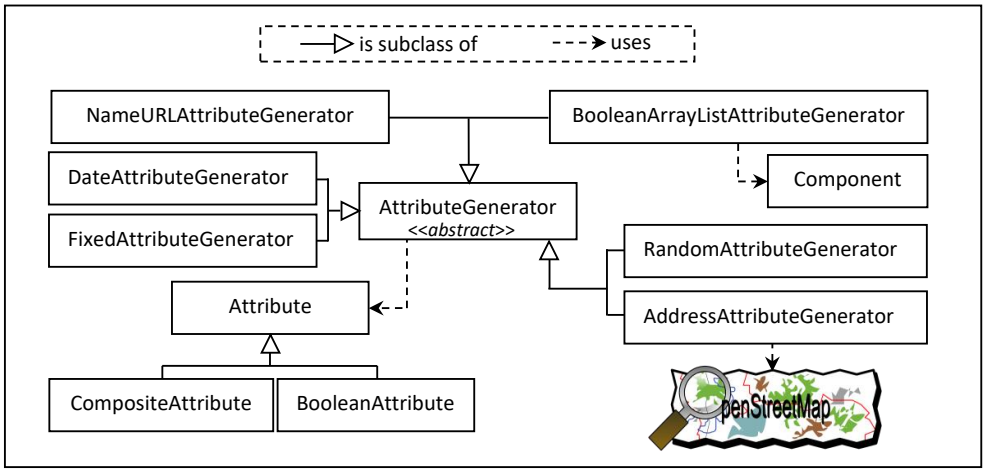


Figure 5.4: Class diagram for attribute generation in DataGenCARS.

Moreover, it is possible to extend the existing classes to simulate any behavior desired. Thus, any other required generator can be easily plugged in by extending the abstract class *AttributeGenerator* (or any of its subclasses) and defining the desired Java code in the implementation of the method *generateValueAttribute* (abstract method in *AttributeGenerator*). In the input files defining the different schemas, it is possible to indicate also potential input parameters that may be required by such generators, such as references to external data sources or files, or specific values to consider for the generation of attribute values (e.g., see input_parameter_attribute_1 in Figure B.2 (in Appendix B.1.1). In this way, we combine the flexibility of Java to define the desired behavior with an easy modification of the schema files to adjust that behavior in the way desired.

InstanceGenerator. This class generates instances of users, items, and/or contexts for the output dataset, by combining values generated from the different attribute generators specified (see Figure 5.5). The power of the *Java Reflection API* is exploited to generate the required instances dynamically by interpreting the information provided in the input schema files. Besides the schema file, this class interprets

the input files describing the user and item profiles, as well as the input file defining the generation parameters desired.

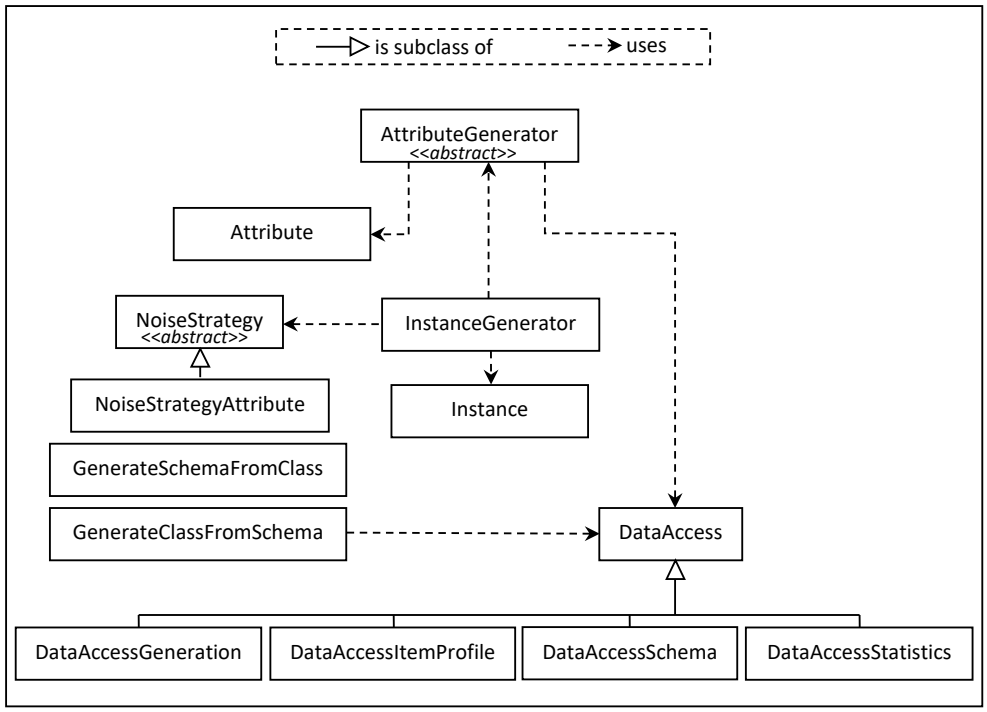


Figure 5.5: Class diagram for instance generation in DataGenCARS.

RatingGenerator. This class generates user ratings. For that purpose, it obtains random combinations <user, item, context> and computes the corresponding rating by applying the data utility function defined in the user profile of the corresponding user. To compute the rating, this class takes into account whether the user profile defines a preference for higher or lower values in the domain of values of each attribute and transforms each attribute value into a normalized score in the output range required (e.g., a value between one and five), through linear regression.

As real users do not rate items by applying a precise mathematical function, potential noise or uncertainty for the rating generated can be simulated by defining a weight different from zero for the attribute *others* in the user profile. Moreover, this class generates a random time and date (within the allowable input range of dates provided) representing the time instant when the user provided the rating; random timestamps are generated in such a way that the ratings of the users are mixed with each other, so supporting situations where several ratings are provided by the same or by different users at approximately the same time.

As it can be observed in Figure 5.6, RatingGenerator is an abstract class with two concrete implementations that add the specific adaptations required for two dif-

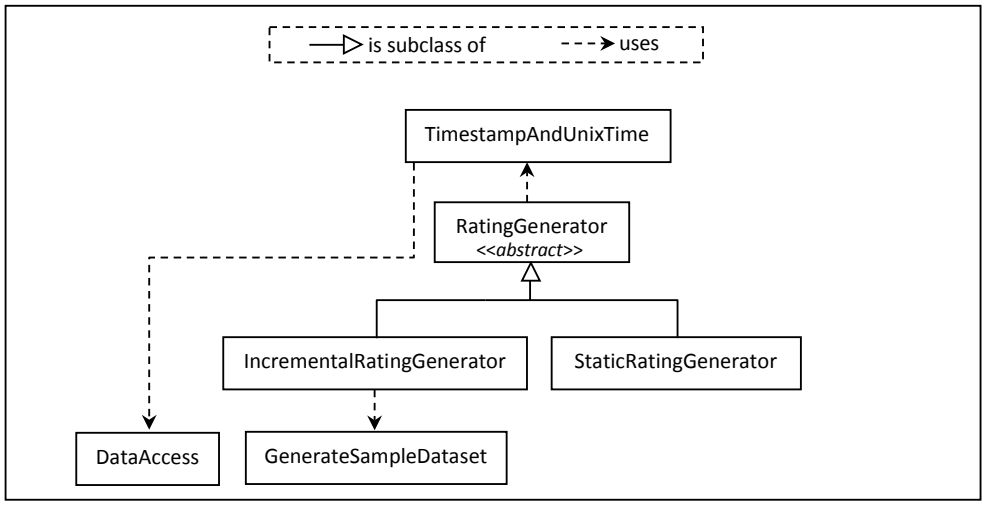


Figure 5.6: Class diagram for rating generation in DataGenCARS.

ferent use cases: *StaticRatingGenerator* generates a fixed number of ratings required, while *IncrementalRatingGenerator* supports the incremental extension of an already-existing dataset.

DataAccess. As depicted in Figure 5.7, this class, along with its related classes *DataAccessSchema*, *DataAccessGeneration*, *DataAccessItemProfile*, and *DataAccessStatistics*, is in charge of interpreting the input files and extracting the values needed during the dataset generation process applied by DataGenCARS.

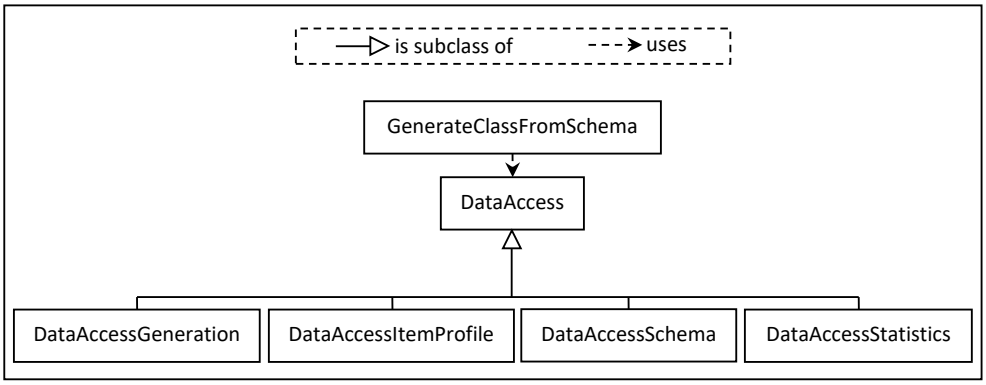


Figure 5.7: Class diagram for data access in DataGenCARS.

ExtractStatistics. This class obtains statistical information about users, items, contexts, and ratings, from an input dataset. As it is shown in Figure 5.8, it contains two class specializations: *ExtractStatisticsUIC* (statistics about users, items and con-

texts) and *ExtractStatisticsRatings* (statistics about ratings). These classes use the *StatisticalMeasure* class, that computes average, variance, and standard deviation measures. Moreover, other statistical characteristics are also obtained, such as the number of identifiers, the number of possible values by variable, the frequency or percentage of values by variable, the number or percentage of ratings by user, the percentage of items and contexts repeated by user, and the number of users, items, contexts, and ratings.

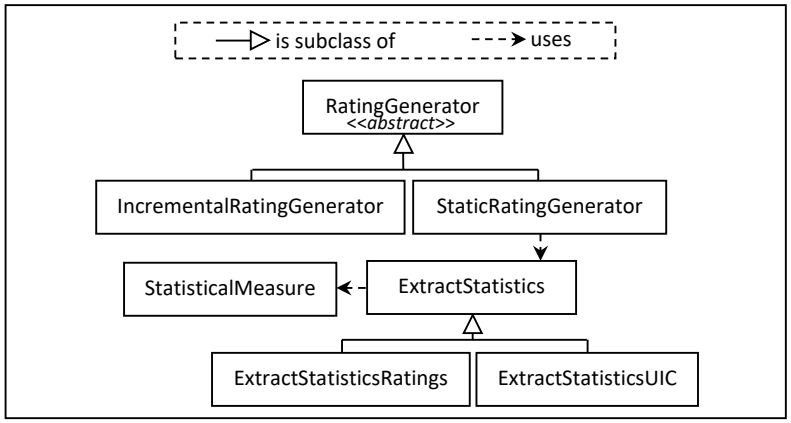


Figure 5.8: Class diagram for the statistical extraction process in DataGenCARS.

5.2 DataGenCARS in Action

In this section, we illustrate some capabilities of DataGenCARS by exploring typical tasks that can be performed by using it. Before entering into technical details, Figure 5.9 provides some usage guidelines that facilitate understanding the functionality of DataGenCARS at a glance. When a researcher faces the problem of finding a way to evaluate a recommendation algorithm, she/he also needs to tackle the issue of finding out which data can be used to carry out such evaluation. The easiest way for evaluating a recommendation algorithm is to use an already-existing dataset that fulfills the expectations of the researcher (e.g., it includes context data, it has enough ratings for the results to be significant, the domain is similar to the one the algorithm has been designed for, and so on). Many times, existing datasets do not fully cover all the features we are interested in, so a new dataset should be created. For this purpose, we distinguish three main lines of action:

1. Perform a survey/study with real users or develop an app to capture the data needed to build the dataset.

2. Design and implement an attractive game to gather information to build the dataset through the user interaction with the game (gamification).

3. Build a synthetic dataset with the help of some tool.

We have designed DataGenCARS for this last case. It not only supports building datasets from scratch by using different parameters indicated by the user but also creating datasets from existing collections of data, either enlarging or completing them (e.g., to avoid potentially missing features such as context data or a low number of ratings). In the rest of this section, we detail different functionalities provided by DataGenCARS in a more technical fashion.

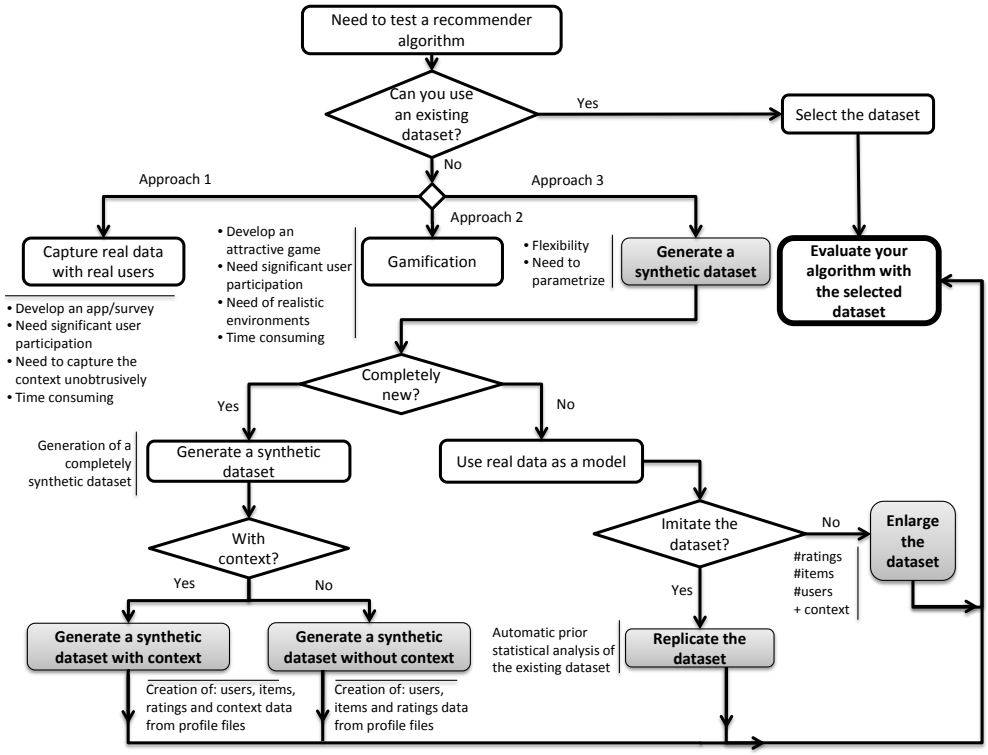


Figure 5.9: Basic guidelines for the evaluation of CARS.

5.2.1 Generation of a Synthetic Dataset Similar to an Existing One

Figure 5.10 shows a workflow whose purpose is to generate a synthetic dataset that exhibits properties similar to those present in a real pre-existing dataset. As shown in the workflow, in some cases (e.g., for the LDOS-CoMoDa dataset described in Section 8.3.2.2), we may need first to split an original dataset file that contains in a single file all the information (information about users, items, contexts, and ratings)

into several files (shown in Figure B.1 in Appendix B.2). For this purpose, this step must be provided with the required splitting rules that delimit the information related to the users, items, contexts, and ratings (e.g., the positions of attributes in each row of the input file that correspond to each of those types of data).

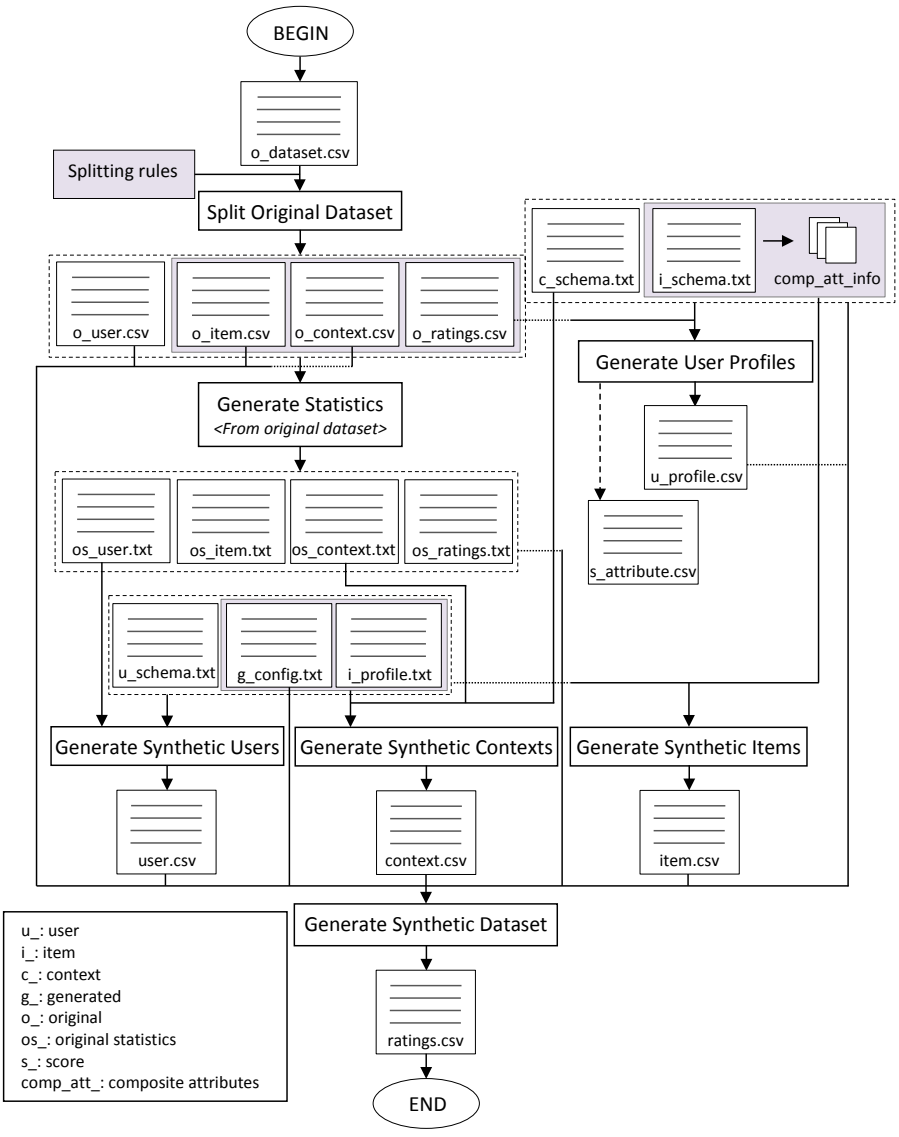


Figure 5.10: Workflow to generate a synthetic dataset similar to an existing one with DataGenCARS.

Besides, in order to obtain a synthetic dataset with value distributions similar to those of a real dataset, DataGenCARS first needs to obtain statistics of the output files obtained in the previous step. The statistics obtained are stored in several output files that contain statistical information about the users, items, and contexts. This statistical information includes the number of different identifiers, as well as information related to the remaining attributes (e.g., number of possible values, frequency of occurrence of their values, average, and standard deviation). Moreover, the collected statistics about ratings contain information such as: the total number of users, items, contexts, and ratings; the number and percentage of ratings per user; the number and percentage of different (i.e., not repeated) items and contexts per user; the percentage of equal (i.e., repeated) items and contexts per user; the average and variance of the number of items per user; and the average and variance of the number of contexts per user. By default, DataGenCARS will capture all the statistical information available. However, it is also possible to inspect and manually modify the statistics gathered by DataGenCARS as desired; for example, an expert user could inspect the statistic files generated and manually remove rarely-occurring features in the original dataset (e.g., if she/he considers some values in the original dataset to be noise). In addition, DataGenCARS facilitates the automatic inference of utility functions (or user profiles) from a real dataset, by using a class called *GenerateUserProfile*. At the end of Section 5.1.1, we explain the details about the automatic learning of user profiles from an existing dataset.

5.2.2 Generation of a Completely-Synthetic Dataset

Figure 5.11 illustrates the steps followed to generate a dataset that is completely synthetic. The steps are analogous to those presented in Section 5.2.1 concerning the generation of a dataset similar to an existing one (see Figure 5.10). The main difference is that in this case we do not have an existing dataset to use as a starting point. Therefore, rather than obtaining statistics of data distributions from an existing dataset, the workflow relies on manually-defined configuration files, user profiles, and schemas.

5.2.3 Generation of a Dataset of Ratings Incrementally

Figure 5.12 shows the workflow corresponding to the incremental generation of a dataset of ratings: given an input dataset (real, synthetic, or partially synthetic), it generates another dataset with a larger number of ratings. The new ratings generated are obtained by applying the utility functions that define the user profiles, which could be defined manually or learnt from an existing dataset (as explained in Section 5.2.1). If needed, an expert user could also adjust manually the utility functions automatically obtained.

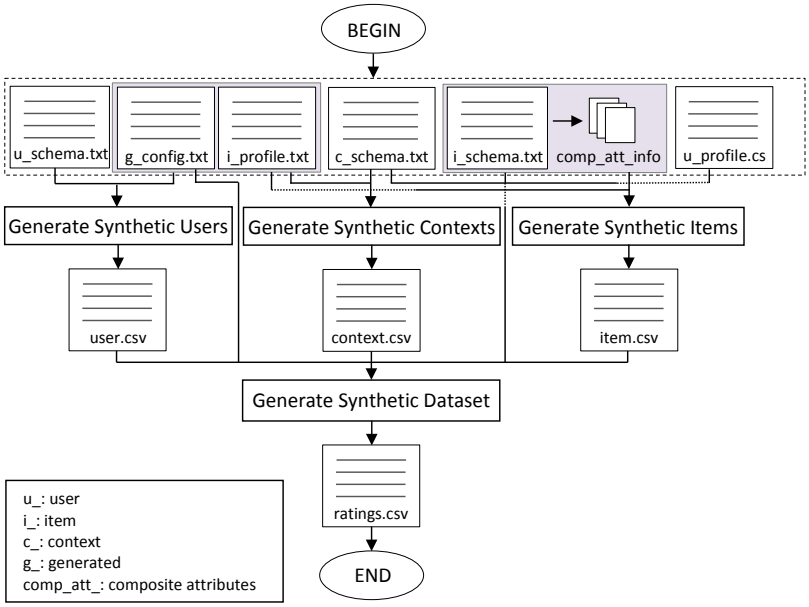


Figure 5.11: Workflow to generate a completely-synthetic dataset with DataGenCARS.

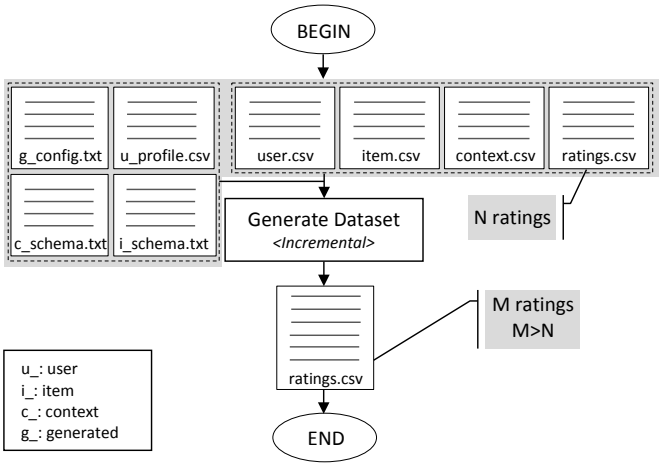


Figure 5.12: Workflow to generate a dataset of ratings incrementally with DataGenCARS.

5.2.4 Completion of Missing Information in Datasets

Sometimes, existing datasets for the evaluation of CARS contain unknown values related to the items and/or the contextual information of the ratings. This is particularly true in cases when the user himself/herself has to manually enter information about her/his context when providing a rating. Thus, most users will feel overwhelmed with that need and provide only basic information, leaving most input fields empty. Fortunately, DataGenCARS allows replacing the unknown values in a dataset by useful information, by using the *ReplaceUnknownValues* class. It is an interesting feature because it would allow evaluating context-based recommendation algorithms on context-rich datasets that are not completely synthetic.

Figure 5.13 shows the workflow followed by DataGenCARS to replace unknown values in the item and context files. For both files, DataGenCARS first checks, for each instance, whether the value of the current attribute is *null*. If so, it generates a new value (other than *null*) by using the corresponding schema file. Then, the value *null* is replaced by the new value. Otherwise, the current value of the attribute is preserved.

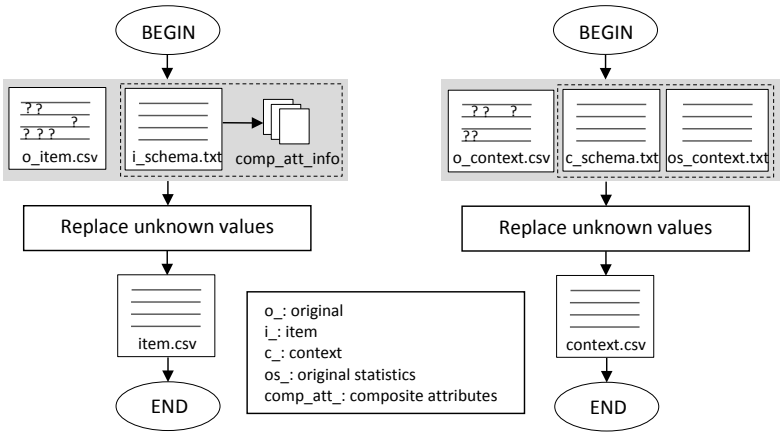


Figure 5.13: Workflow to complete unknown contextual information with DataGenCARS.

5.2.5 Composition of Workflows

To conclude this section, it is important to emphasize that the previous ones are just some examples of tasks that can be performed by using DataGenCARS. However, the possibilities are much numerous. For example, by using DataGenCARS we can also introduce in a dataset uncertainty in the ratings or unknown values. As another example, we can generate a synthetic dataset based on a subset or sample (seed) of an original dataset (as shown in Figure 5.14), by considering as a basis only a subset of the original ratings.

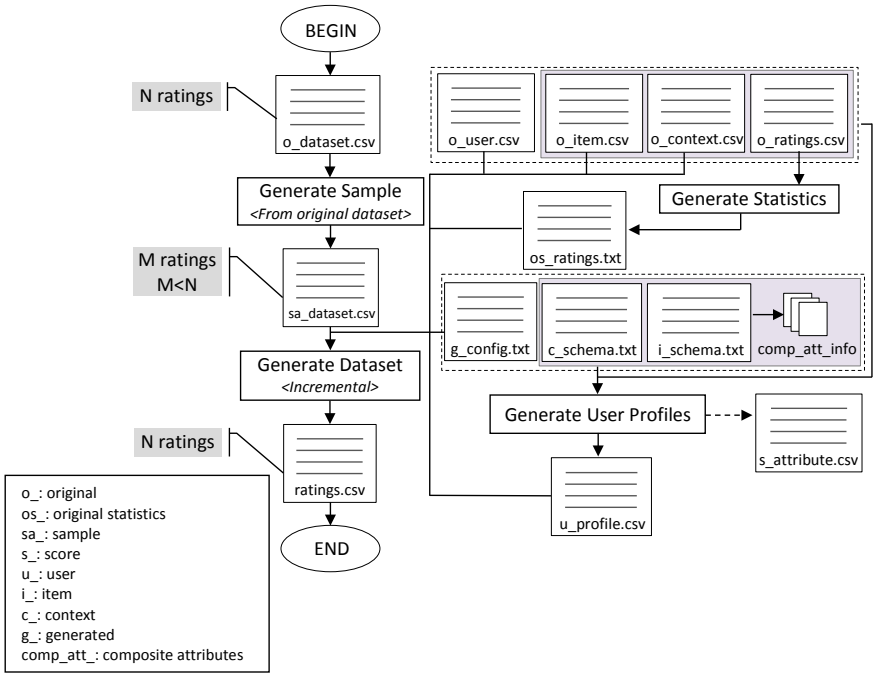


Figure 5.14: Workflow to generate a dataset from an initial sample of an existing dataset with DataGenCARS.

Besides, we could even chain workflows or combine parts of several of the workflows presented to generate datasets that suit our needs. For example, Figure 5.15 illustrates a workflow that generates a synthetic dataset and then increments its number of ratings. We could also use (part of) the workflow presented in Section 5.2.1 to learn user profiles from an existing dataset and then apply those profiles to recompute ratings in a different synthetically-generated dataset.

5.3 Real Data vs. Synthetic Data

Real datasets, even if they are available, can be subject to important limitations, such as bias or incomplete data. For example, there could be a temporal bias if the recollection process focuses on a specific time period and the ratings exhibit seasonality trends. Similarly, there could be a context bias if ratings are collected only in certain contexts. The aforementioned example with the STS dataset shows that it may be usual to have missing context data (e.g., see the analysis of the STS dataset presented in Section 8.2.1), which could be a problem if the goal is to evaluate algorithms for CARS. Besides, with a real dataset it is possible to evaluate recommendation algorithms only under the specific conditions represented in the dataset. On the other

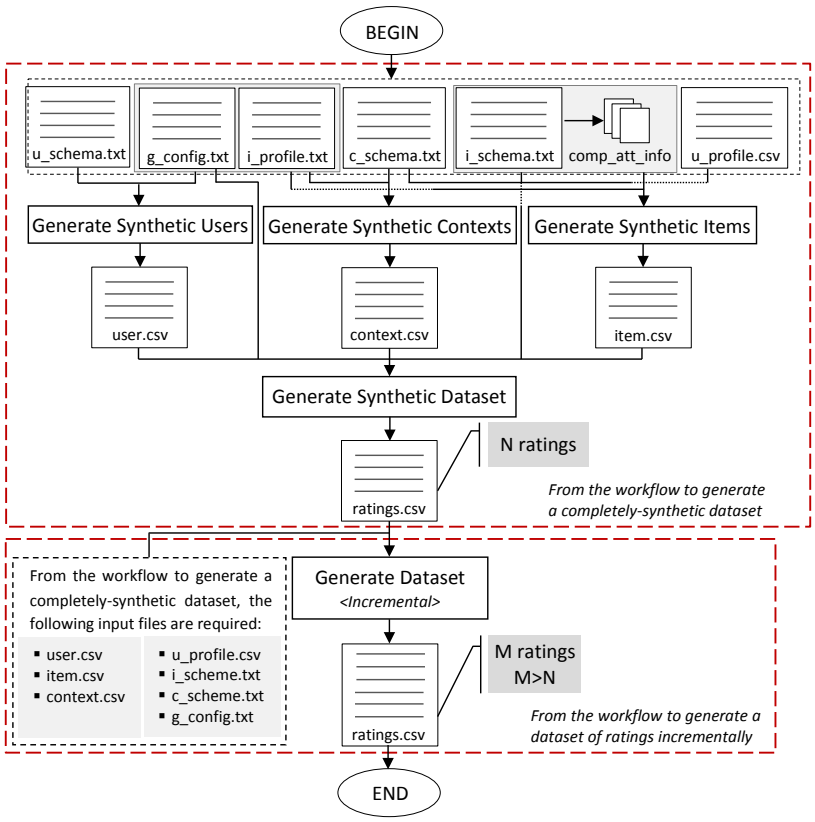


Figure 5.15: Example of composition of workflows for DataGenCARS.

hand, by generating appropriate synthetic datasets we can evaluate recommendation approaches in any desired condition, including usual cases that could frequently be observed in reality (reproducing expected situations, or real observations, in the synthetic data generated) and also extreme cases that are more rarely expected. So, real data may guarantee *veracity* (if the data collection process is reliable) but not necessarily *validity* for our evaluation purposes. With DataGenCARS we can generate artificial data that is valid for the targeted purposes. The difficulty lies in the need to specify the features of the dataset desired. However, DataGenCARS allows considering multiple scenarios, representing different hypothetical situations, for example to determine how a certain recommendation approach would behave if little context data is available, if the distribution of ratings among users is highly non-uniform, or if users tend to over-rate items.

On the other hand, when a synthetic dataset is generated, some steps should be performed to try to ensure that the generated dataset satisfies the purposes it has

been created for. For example, if the synthetic dataset has been built based on real observations (i.e., there is a previously-existing dataset), it is interesting to compare the behavior of the recommendation algorithms we want to evaluate in both the synthetic data and the real data, as we have done in experiments presented in Section 8.3.2. Nevertheless, it should be noted that a full fair comparison might not be possible in the general case, as for example the number of real observations may be insufficient for the recommendation algorithm to perform well with the real data available, feature variations may have been introduced to fit the data to the evaluation purposes, new attribute values (e.g., context data) might have been added, etc. In other cases, there are no real observations at all, and therefore the only way to evaluate the synthetic dataset is by verifying the desired statistical properties. As a summary, Figure 5.16 shows some basic tasks that can be performed to evaluate synthetic datasets; it should be noted that, for simplicity, the performance metrics shown in the figure are related to the prediction accuracy of the recommendation algorithms, but any other relevant evaluation metric could be considered [SG11, HKTR04].

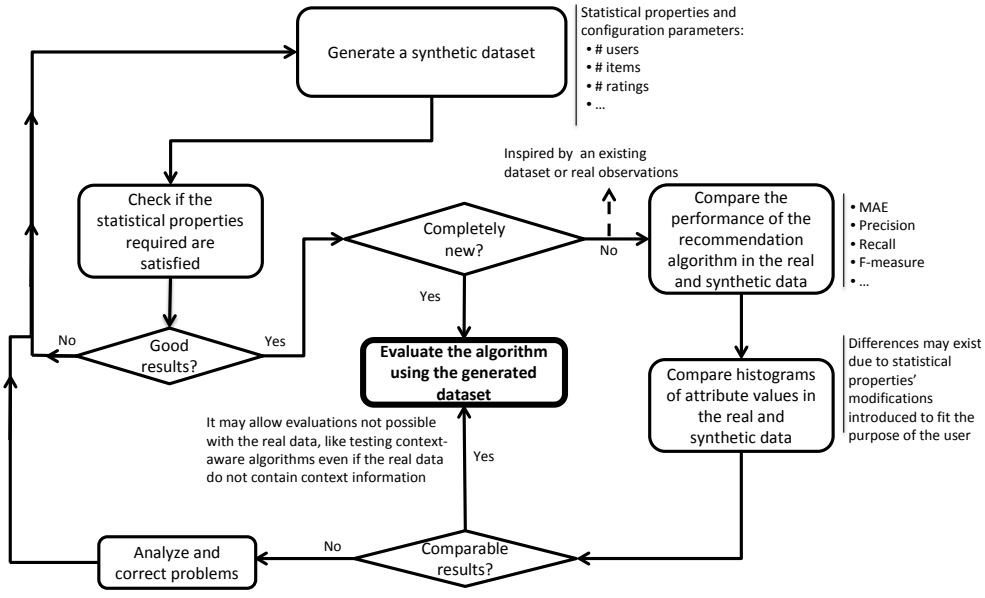


Figure 5.16: Basic guidelines for the evaluation of synthetic datasets.

5.4 Summary of the Chapter

In this chapter, we presented DataGenCARS, a tool for the automatic generation of datasets that is appropriate for the evaluation of context-aware recommendation algorithms. First, we described the main functionalities of DataGenCARS, such as flexible definition of schemas, realistic generation of ratings, consistent generation of

attribute values for items, generation of mixed real-synthetic datasets, support for the automatic mapping between item schemas and Java classes, and automatic learning of user profiles from an existing dataset. The tool is very generic and can fit different application domains and sets of needs, by appropriately defining a set of schemas as input files that will direct its behavior. Second, we explained the most important classes of its architecture. Third, we presented the different workflows supported by DataGenCARS. Finally, we summarized the basic guidelines for the evaluation of synthetic datasets. With the tool, we can simulate any desired situation as well as reproduce or mimic any existing one (e.g., generate new datasets with features similar to other existing ones). Therefore, DataGenCARS represents an interesting tool that can help alleviate the problem of scarcity of datasets suitable for the evaluation of recommendation strategies, especially those that require rich context information.

Later, in Section 8.3, we will discuss an evaluation experimental of DataGenCARS. A first set of experiments illustrates how the synthetically-generated datasets can be used to evaluate different types of recommendation algorithms in a variety of situations. They show how the flexibility of DataGenCARS supports the generation of different datasets with different features, adapted to the evaluation needs. For example, we can easily generate the desired amount of context information, which is a key element to evaluate context-aware recommendation algorithms. Moreover, a second set of experiments shows how, by appropriately setting DataGenCARS, we can generate any dataset required, including data similar to other existing observations that correspond to situations observed in the real world. Summing up, we can generate any dataset desired, either inspired by other real datasets or not, and so also datasets that do not exist or would be very difficult to obtain.

Chapter 6

Use Case Scenario: Recommending Items in a Museum

In this chapter, we present a use case scenario: the recommendation of items to observe in a museum. In Section 6.1, we introduce a brief summary of the state of the art on recommendation systems for museums as well as a motivation for this use case. In Section 6.2, we describe the specific scenario considered. In Section 6.3, we present the prototype developed for a subsequent experimental evaluation (shown in Section 8.4). Besides, we discuss the process of synthetic generation of a rating collection provided by users about the works of art that they observe in specific contexts, by using DataGenCARS (see Chapter 5).

6.1 Recommendation Systems for Museums

Providing appropriate recommendations for museum visitors has been a hot topic in the field of recommendation systems, and so we can find a significant number of works dealing with it. Some of them focus on the design of systems that serve as the visitors' guide trying to offer a personalized experience. For instance, a couple of collaborative models to predict the upcoming locations of users in a museum have been proposed [BZB+08], which could be used to build user models; more specifically, a temporal approach to predict future locations based on the potential interest of unseen exhibits, as well as a transitional approach to predict future locations based on the paths followed by other visitors, are presented. A collaborative filtering algorithm to recommend trajectories to visit POIs, which is evaluated with data regarding the Vienna Zoo (Austria), has also been presented [HG11]. As contextual information, this system takes into account the locations of the users, and so the authors present a notion of similarity between contexts. Although this work presents rather inter-

esting features, it has also some drawbacks (e.g., it does not consider the opinions of other visitors, in terms of ratings provided rather than just POIs visited, and its evaluation is quite limited). Regarding the use of collaborative filtering to recommend attractions to visitors, we also find an appealing approach about POI recommendation taking into account the location diversity in [CZC+15]. This work is evaluated by using outdoors check-in app-related datasets where the only contextual information provided is the location of the users. In our opinion, those datasets somehow limit the different types of scenarios where recommender systems could be tested on, especially when dealing with CARS, as the context information available should be rich instead. *MusA* [RXM+13] is a generic framework to develop multimedia guides for mobile devices and provides a vision-based indoor positioning system, as well as thematic paths created by professional curators or museum staff. Another interesting work is *SmARTweet* [CMMP13], a location-based application able to detect visitors' nearby artworks and show their corresponding history using multimedia resources. It also uses collaborative filtering to recommend personalized artworks based on the visitors' interests (gathered via questionnaires) and behavioral information collected by tracking the users throughout the museum.

There exist some other approaches also aimed at building reliable guides for visitors in museums. For example, *UbiCicero* [GPSS09] provides recommendations thanks to the detection of artworks nearby by using RFID readers assembled in mobile devices. It also takes into account some contextual information, such as the visitor's current position and her/his behavioral history. With this information, the system returns personalized recommendations. The evaluation of this proposal is quite limited too, since the system is tested with groups of 5-7 users. The recommendation system for mobile devices for museum visitors presented in [BL15] can be seen as a step forward, as it is able to adapt to the user's interests, by taking into account contextual information such as the location, level of expertise in art, and time. It uses a hybrid mechanism based on collaborative filtering combined with a post-filtering semantics-based approach.

When focusing on the way these systems are tested, we realize that there is no common methodology. To our knowledge, there does not exist any reliable dataset for testing CARS, especially in indoor environments. Thus, some works deploy real implementations and test the suitability of the systems by asking the visitor to fill out questionnaires when leaving the museum, in order to know their satisfaction [GPSS09, RXM+13, RHS+13, AAP+14]. Although this method might be a good way to evaluate this type of systems, the real outcomes seem to be very simplistic, since collecting a significant and reliable set of responses is too costly; moreover, even if the visitor is satisfied, it could be difficult to know if she/he could be even more satisfied with a different set of recommendations. We can also find some approaches that do not test their approach but present different case studies [TCL08, CHGS05, JKSS07, BL15].

6.2 Description of the Use Case

Motivated by the shortcomings mentioned in Section 6.1, we decided to study the problem of recommendations in a museum as a specific case study, which we introduced in [dCRHIHTL17c]. The goal of the recommendation system is to maximize the user's satisfaction with the visit. Specifically, the system will suggest to the user a trajectory to follow within the museum, prioritizing the sequence of works of art to observe in the museum within the visiting time available. By using DataGenCARS along with a simulator of museum visitors (mobile users), we will not be so constrained by the limitations mentioned in Section 6.1 regarding the evaluation of the proposed recommendation system.

In this section, we first describe the main features of the works of art used in the proposed use case. Then, we explain some particularities and generalities of the layout of the Museum of Modern Art (MoMa) in New York.

6.2.1 Works of Art

We use a real dataset corresponding to the works of art in the MoMA [The16]. This collection contains information about 129,024 works of art, characterized by 29 attributes, and belonging to 14,949 different artists. The works of art include basic metadata, such as the title of the work, its dimensions, its author and her/his nationality, etc. In order to facilitate the management of this dataset, we performed some preprocessing tasks. For example, we discretized some attributes, like the date of acquisition by the museum, which was classified into "recent" and "no recent" (as some frequent visitors may feel attracted towards works of art that have been acquired recently), and binarized others (like the nationality of the work of art).

Unfortunately, the precise location of each work of art is not available in the original dataset. However, the location of each item is important for the evaluation of context-aware recommendation systems. Therefore, we used the Java-based synthetic data generator DataGenCARS (see Chapter 5), to create a random location for each work of art, subject to some rules to ensure a feasible real-world distribution: the works of art are distributed among the floors and rooms of the museum, and then positioned equitably along the walls and interiors of the rooms, in order to avoid overcrowded areas or situating two works of art with very little separation between them. For that purpose, a class *ItemLocationAttributeGenerator* was defined to extend the DataGenCARS framework.

6.2.2 Layout of the Museum

According to the documentation available in the Internet [The04], the MoMA museum is composed by six floors (see Figure 6.1):

- *First floor:* it hosts the hall and a garden with sculptures.
- *Second floor:* it contains contemporary works of art.

- *Third floor:* it is devoted to architecture, design, drawings, and photographs.
- *Fourth and fifth floors:* they include paintings and sculptures.
- *Sixth floor:* it is the place of special exhibitions.

We mimicked part of the layout of the MoMA museum. Without loss of generality, for simplicity, we focused on floors fourth and fifth and restricted the types of works of art considered to paintings and sculptures (a total of 240 items were considered). From map images available on the Web, we reproduced the layout of those floors by converting the images to graph structures.

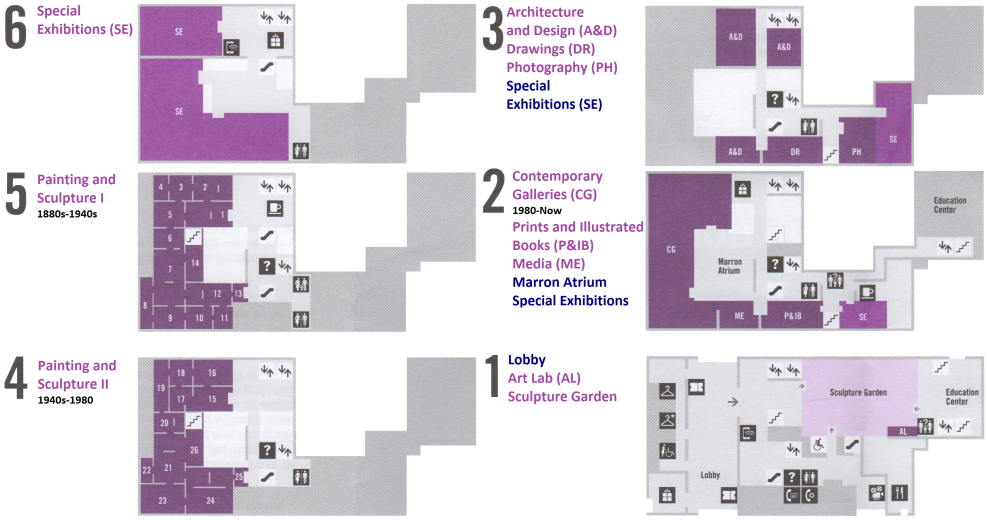


Figure 6.1: Map of the MoMA museum.

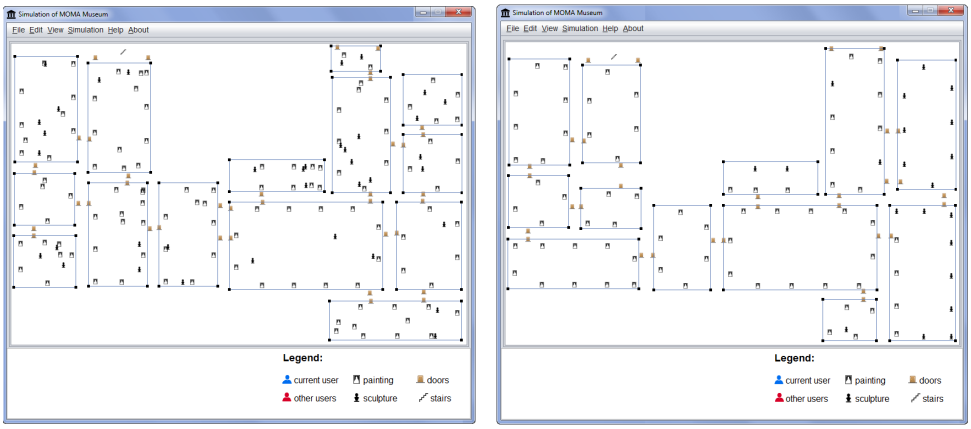
6.3 Prototype Developed and Generation of Synthetic Ratings

We developed a museum simulation application that loads and displays the layout of the selected floors as well as the paintings and sculptures in the area, as shown in Figure 6.2. This desktop application has been implemented by using the tools shown in Table 6.1 and the general recommendation framework presented in Chapter 3.

Figure 6.3 shows a snapshot of the simulation application at a certain time instant, where we can observe different users observing works of art or moving to other locations. In the figure, there are no doors connecting rooms in the left part of the figure and the right part of the figure: the reason is that the left and the right part of the figure correspond to different floors (connected through the stairs) but, for the sake of simplicity we show them next to each other.

Tool/library	Use
Java 8	Programming language
Swing	GUI widget toolkit for Java
WebPlotDigitizer [Roh07]	Obtention of key locations (e.g., rooms, doors, and stairs) from raster map images of the museum
JGraphX [Cha16]	Management and visualization of graph structures
Sqlite-jdbc [Sai16]	Management of SQLite databases

Table 6.1: Tools used for the implementation of the museum simulation application.



(a) Fifth floor (b) Fourth floor

Figure 6.2: Simulation application showing the map of the MoMA museum.

The MoMA dataset contains only information about the works of art in the museum, not about ratings provided by users. Therefore, we used DataGenCARS to generate rating data enriched with context information, that is, a set of ratings (opinions) provided by users about the works of art that they observe in specific contexts. For this purpose, we defined different *user profiles*, which are scoring functions that are used exclusively to synthetically generate a rating based on a set of attribute values [dCRHIHTL17a], and assigned a specific user profile to each user simulated. These profiles are obviously unknown to the recommendation system and include context attributes that might affect the ratings provided (see Table 6.2). For each user, and based on her/his profile, we generated a rating for each work of art in the museum, in each possible context. Besides, for each piece of art, we generated a synthetic attribute representing the emotion transmitted by that piece (happiness, sadness, or

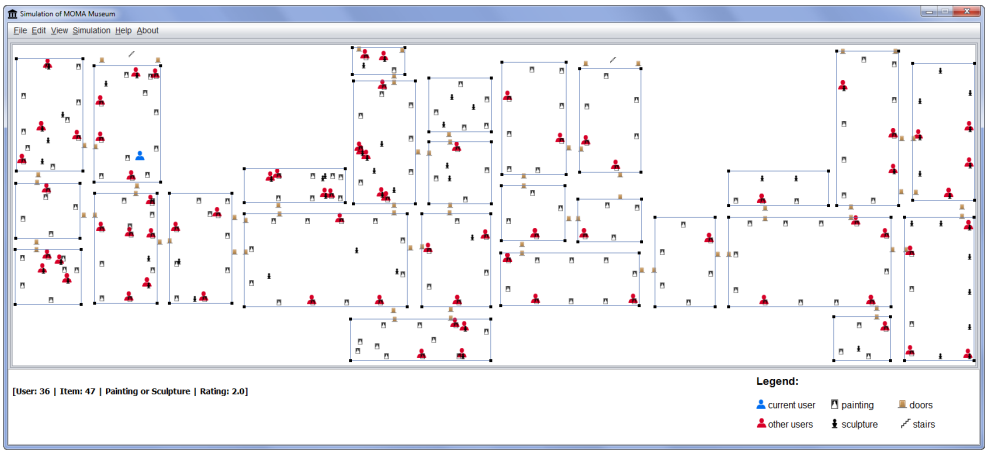


Figure 6.3: Simulation application in action: visitors in the museum.

neutrality) and defined a simple rating adjustment function that can slightly modify the rating generated depending on the relation between the current mood of the user and the emotion transmitted by that work of art.

Context attribute	Possible values
User's mood	Happy, sad, or neutral
Temperature of a room	Warm, hot, or cold
Number of people in a room	Large, medium, or small
Noise level in a room	High, medium, or low

Table 6.2: Context attributes considered in the museum use case scenario.

The desktop simulation application developed supports the simulation of visitors moving throughout the museum, observing works of art, and releasing ratings. When a visitor's trajectory stops at a certain work of art, she/he observes it for a certain amount of time and then emits a rating according to her/his satisfaction with that work and the current context. As commented before, the ratings were previously generated using DataGenCARS, and the simulator just needs to select the correct one according to the specific context existing at that time (e.g., considering the current mood of the user).

6.4 Summary of the Chapter

In this chapter, we first presented the motivation that led us to develop the proposed use case. Then, we described the main features of our use case. Finally, we presented the prototype developed in order to evaluate context-aware dynamic recommendations

in real-time about items to observe in a museum. While describing the prototype, we explained the process followed to build a mixed scenario using both real data (data about work of arts and map information) and synthetic data (ratings provided by the users about the work arts in specific contexts), by using DataGenCARS (described in Chapter 5). In this way, we contributed to filling the gap between the problems of design and evaluation of solutions for CARS.

Later, in Chapter 8, we will present an experimental evaluation performed on this use case. The experimental results that we will show in Section 8.4 show that it is possible to deploy a trajectory-based recommendation strategy based on collaborative filtering, by using a mobile P2P ad hoc data sharing solution to exchange directly data among the mobile devices of visitors located nearby.

Chapter 7

Related Work

We already indicated in Section 2.3 the main features and challenges of context-aware recommendation systems, as the new generation of recommendation systems. In this chapter, we present work related to our proposal in the field of CARS. In Section 7.1, we first review works of context-aware recommendation systems that assume a static environment. In Section 7.2, we focus on context-aware recommendation systems for mobile environments. We consider mobile context-aware recommendation systems to those works where the authors make it clear that they exploit dynamic context information and the mobility of the users. In Section 7.3, we indicate some works of location-aware recommendation systems, as a specific case of context-aware recommendation systems. In Section 7.4, we present some mobile recommendation approaches based on P2P networks. Finally, in Section 7.5, we analyze some popular synthetic data generation tools, which are not suitable to generate datasets for the evaluation of context-aware recommendation approaches, and so they motivate the development of DataGenCARS.

7.1 Approaches for Context-Aware Recommendation Systems

In this section, we first present some works that propose specific algorithms for CARS, which are complementary works to the one developed in this thesis. These algorithms could be easily integrated into the proposed architecture and evaluated using synthetic data obtained by using DataGenCARS. Then, we describe existing frameworks to facilitate the development and evaluation of CARS. Finally, we highlight some examples of context-aware recommendation systems in different application domains.

7.1.1 Algorithms for CARS

Several studies have proved that contextual information improves both the quality of the recommendations and the user satisfaction when a recommendation system is

employed [ADNZ15]. For example, the authors of [ASST05] proposed that the recommendation procedure should incorporate contextual information. In [AMRT11], the authors classified the context-aware recommendation approaches into pre-filtering, post-filtering, and contextual modeling paradigms. This classification facilitated the organization and understanding of the emerging context-aware recommendation approaches.

7.1.1.1 Contextual Pre-filtering Algorithms

Several approaches based on contextual *pre-filtering* have been proposed by the research community of CARS. An example is the reduction-based pre-filtering approach proposed in [ASST05]. It strictly uses the ratings generated in the same user's current context to learn a 2D collaborative filtering model that will predict unknown ratings. A limitation of this approach is its rigidity in the selection of contextualized ratings. Hence, the authors studied the possibility of pre-filtering generalization, by aggregating ratings with hierarchically-related contextual conditions.

In order to reduce the computational cost, and motivated by the idea that an item can have different evaluations in different contexts, an interesting contextual pre-filtering approach for collaborative filtering, called *Item Splitting*, was presented in [BR14]. It dynamically discovers, for each item, the relevant contextual factors. In this method, the set of ratings for an item is not filtered but it is split into two subsets taking into account the value of a context variable. These two sets of ratings are then assigned to two new virtual (or fictitious) items, which are used in the prediction algorithm instead of the original one. This split is performed only if there is a statistically significant difference between the ratings for this item in two alternative contextual conditions. The experimental results show that this method provides more accurate rating predictions than the reduction-based pre-filtering approach proposed in [ASST05]. Item splitting was previously introduced in [BR09a, BR09b]. A similar pre-filtering approach was presented in [AM06], but in this case the contextual information was used to alter the user model. In [Zhe13], a combination of item splitting and user splitting, called user-item splitting (UI splitting) was presented.

An empirical comparison of context-aware splitting approaches (e.g., item splitting, user splitting, and UI splitting) on different datasets (food and movies) was performed in [ZBM14]. The experimental results showed that splitting approaches outperform several popular context-aware CF algorithms. However, among the different splitting approaches there is no clear winner. Recently, another empirical evaluation of context-aware splitting approaches in a movie dataset was explored in [FPH$^+$16]. In this comparison, the UI splitting approach outperforms the other context-aware splitting approaches.

The data-sparsity problem of CARS was tackled in [CRC16], by exploiting semantic similarities (based on distributional semantics [RG65, MSLdG14]) between contextual situations in a reduction-based pre-filtering approach. During the filtering process, this method considers in addition to the ratings generated by the user exactly in a specific situation, those generated in similar situations. This approach was previously introduced by the same authors in [CRC13a, CRC13b].

7.1.1.2 Contextual Post-filtering Algorithms

Approaches based on contextual *post-filtering* have also been proposed in the literature of CARS. For example, two probabilistic post-filtering approaches (weight and filter) were considered in [PTG+09]. The first approach reorders the recommended items by weighting the predicted rating according to their estimated probability of relevance in the existing context. The second approach filters out the recommended items that have a probability of relevance lower than a specific threshold.

A post-filtering approach was also proposed for the movie domain in [CGQT11]. The proposed contextual recommendation approach, called *Context-RS*, differs from the traditional post-filtering approach in the adopted item selection technique. In Context-RS, the items obtained from a traditional recommendation model are filtered by using association rules to find useful correlations between the current context and the features of each of the candidate items. The experimental results presented in [CGQT11] show that Context-RS outperforms traditional recommendation methods.

In [LKH14a], the authors used the contextual post-filtering paradigm in a *smart TV* system, which recommends multimedia content. The contextual recommendation approach first applies a social tag-based CF, and then the resulting recommendation ranking is adjusted by taking into account context information.

Recently, a context-aware recommendation algorithm with two-level SVD, called *CTLSVD*, was proposed in [CHY+17]. CTLSVD is a contextual post-filtering approach that filters the recommendation results of the SVD algorithm, by using the current time as contextual information. For two movie datasets, the experimental results show that CTLSVD provides more accurate recommendations compared with traditional methods.

7.1.1.3 Contextual Modeling Algorithms

Many of the *contextual modeling* approaches are based on extending *Matrix Factorization* (MF) techniques [KBV09]. Generally, MF techniques learn latent features of users and items from the seen/known ratings in the 2D rating matrix ($User \times Item$). Then, the unknown ratings are predicted by using these latent features. The emerged extensions are due to the fact that MF is not flexible enough to easily add contextual dimensions in the recommendation models. However, these extensions benefit from most of the advantages of MF, such as scalability and accuracy.

Contextual Modeling Approaches Extending MF Techniques: Joint Matrix Factorization

An extension of the classical MF technique to incorporate additional context information (e.g., mood tags for the recommendation of movies) was presented in [SLH10, SLH13]. For that purpose, the authors proposed a *joint matrix factorization* model that, in addition to factorizing the 2D rating matrix (similar to the traditional CF), uses contextual information in the recommendation process as a regularization term.

They also presented a novel similarity measure based on movie mood tags (e.g., sad, anxious, upset, and scared) to relate the items (movies) with the contextual information (tags). According to experimental results in the movie domain, the proposed joint matrix factorization approach achieves an improvement (in terms of precision) compared with the performance of other traditional recommendation approaches.

Contextual Modeling Approaches Extending MF Techniques: Tensor Factorization

Another generalization of the MF technique was the *Tensor Factorization* (TF) approach to context-aware collaborative filtering, called *Multiverse Recommendation*, presented in [KABO10]. The main idea of the Multi-Dimensional TF approach for CARS is to model the context variables in the same way as the users and items are modeled in 2D MF techniques, by considering the interactions between users, items and contexts. For example, the 2D matrix ($User \times Item$) can be converted into a MD tensor matrix ($User \times Item \times Context_1 \times ... \times Context_k$), that includes several contextual factors. In this option, the interactions between users, items and contexts are represented as latent factor vectors. Moreover, the rating tensor is factored into a lower-dimension vector space. For the tensor decomposition, the authors used the *High Order Singular Value Decomposition* (HOSVD) model [LMV00], that requires a dense matrix. However, the model of regularized TF introduced by the authors avoids this limitation, by optimizing only the observed values in the rating tensor. In terms of accuracy, the experimental results show that Multiverse Recommendations outperforms non-contextual MF approaches. Unlike other context-aware recommendation algorithms (e.g., pre-filtering and post-filtering), the proposed TF approach does not require a rating filtering, since it uses all available ratings. Another advantage is its computational simplicity and the ability to manipulate MD.

The generic TF approach, included in Multiverse Recommendation, has two limitations: its model complexity (the number of model parameters to be learnt grows exponentially with the number of context variables and polynomially with the size of the factorization), and it only works for categorical context variables. In [XCH+10], a *Bayesian Probabilistic TF* (BPTF) algorithm that tries to address these limitations is presented. BPTF is able to learn the temporal evolution of online shopping preferences. Besides, in terms of scalability, the authors developed an efficient *Markov Chain Monte Carlo* procedure. The experimental results with several real-world datasets show the advantage of a temporal model (that includes the third dimension of time in the form of a tensor) over non-temporal models.

A second solution for the previous drawbacks was presented in [RGFST11], where the authors proposed to exploit *Factorization Machines* (FMs) [Ren10] to model contextual information and to predict the context-aware ratings. Context-aware FMs can be applied in different context domains (including categorical or real-valued domains). Besides, they provide fast and scalable context-aware recommendations due to the linear complexity of FMs in the number of context variables and the factorization size. An empirical evaluation shows that the proposed context-aware FMs outperform the Multiverse Recommendation in terms of prediction quality and runtime. The pro-

posed model, despite achieving a decrease in computational complexity, it may still suffer scalability problems, if the original user-item rating matrix is huge [LA13].

Contextual Modeling Approaches Extending MF Techniques: Context-Aware Matrix Factorization

Another extension of the classical MF approach, called *Context-Aware Matrix Factorization* (CAMF), to consider the contextual information in the rating prediction of a recommendation model, was discussed in [BLR11]. The authors experimentally analyzed three context-aware recommendation models based on MF (CAMF-C, CAMF-CI and CAMF-CC). These models differ among them in the level of granularity (or complexity) used to represent the interaction between the context and the items. The experimental results with real-world and synthetic datasets show that the best model granularity depends on the domain and amount of data available. However, the proposed CAMF approaches outperform the standard MF approach (without context). In terms of the MAE, the authors also compared the TF [KABO10] and CAMF-CI approaches. CAMF outperforms TF with a real-world dataset, but not with the semi-artificial datasets considered in the experiments. However, both approaches performed better than two other pre-filtering approaches (reduction-based [ASST05] and item-splitting [BR09a]).

Moreover, the combination of social network information and contexts to improve the recommendation performance has attracted great attention recently. An example is the context-aware recommendation system *SoCo*, described in [LA13]. It first applies a random decision tree algorithm [CKY08] to divide the initial rating matrix in rating groups with similar contexts. Then, a matrix factorization is applied to infer the ratings of the items not seen, by using the partitioned matrix. The social network information was introduced in the matrix factorization objective function by including an additional social regularization term to infer a user's preference in accordance with her/his friends' tastes. The user similarity measure was determined with a context-aware version of the Pearson Correlation Coefficient, proposed by the authors. SoCo was compared with two real datasets with a state-of-the-art contextual collaborative filtering method called RPMF [ZFY12]. Both RPMF and SoCo similarly apply a tree-based random partition to split the original user-item rating matrix, by grouping users and items with similar contexts, and then apply MF to the resulting sub-matrices. However, among other differences, RPMF does not consider any social network information. In terms of the MAE and RMSE, both SoCo and RPMF outperform classical MF-based recommendation models (with and without social network information) and traditional memory-based recommendation algorithms (e.g., item/user-based CF). However, SoCo outperforms RPMF.

One limitation of applying tree-based random partitions is that a decision tree can only handle categorical contexts. In order to avoid this drawback, a new context-aware social recommendation model (based on MF), named *CSIT*, was proposed in [LCCT17]. It uses a *Gaussian mixture model* (GMM) to perform user-item subgrouping, so it can support both categorical and continuous contexts. In terms of the MAE and RMSE, the proposed method outperforms SoCo and some state-of-

the-art social matrix factorization methods, such as SoRec (Social Recommendation) [MYLK08], RSTE (Recommendation with Social Trust Ensemble) [MKL09], MFTP (Matrix Factorization Technique with Trust Propagation) [JE10], SoReg (Social Regularization) [MZL+11], and RSTR (Recommendation with Social Trust Relationships) [CZZC13].

Contextual Modeling Approaches Extending Similarity Metrics

A variant to integrate the contextual information into recommendation systems has been the extension of the similarity metric applied. For example, the user-based collaborative filtering was extended in [Che05b, Che05a] to predict ratings not only from opinions of similar users, but also from neighbors with contexts similar to the one of the current user, by using the Pearson Correlation Coefficient. In [AT11], a similar idea was described.

Contextual Modeling Approaches Applying Machine Learning Techniques

From the perspective of machine learning, an SVM-based contextual modeling approach was proposed in [ONMU06, ONMU07]. The authors extended an SVM classifier by including contextual information (e.g., the month, hour, weekday, area type, budget, holiday, partner information, weather, and temperature) in the feature space. They also combined a Context-Aware SVM (C-SVM) with Collaborative Filtering, which led to a method called C-SVM-CF. First, the SVM-based approach is used to compute the similarity between users in a specific context. In this case, two users are similar if the SVM approach classifies the liked and disliked items in a similar way. Then, a memory-based approach (CF) is used to obtain the resulting items to recommend. The SVM-based contextual approach was compared (in terms of satisfaction) with other non-contextual approaches in a restaurant domain, resulting the proposed approach the one of the best performance. In [AZK15], a detailed review of context-aware recommendation works that exploits computational intelligence techniques (e.g., fuzzy sets, artificial neural networks, evolutionary computing, swarm intelligence and artificial immune systems) was presented.

7.1.2 Frameworks for CARS

The CARS community has devoted much research effort to the emergence of new context-aware recommendation algorithms (see Section 7.1.1), but not to the development of generic frameworks that facilitate the creation of context-aware recommendation applications. Hence, the development of generic and flexible architectures for CARS is currently an important challenge to address.

An example is the software framework called *Hybreed* [HLGZ14], that has been proposed to build complex context-aware applications. The framework is based on a quite generic notion of context. An interesting feature is that the user of the framework (a developer of recommendation systems or a researcher working in this field) can use different recommendation algorithms (e.g., item/user-based collaborative filtering,

item-average recommendation, random recommendation, slope-one, content-based algorithms, rule-based recommendations, as well as combinations of them). Besides, it provides recommendations for both individual users and groups of users. It also includes methods to generate context-aware recommendations. Finally, it incorporates data from external data sources (e.g., user's profile information, semantic networks, etc.) and is able to store and manage the information in databases (e.g., MySQL or HSQL). However, there are still elements pending for future work, as identified by the authors, such as developing and providing advanced learning algorithms as part of the framework or the development of a processing engine that supports distributed and asynchronous workflows. Moreover, privacy issues were not addressed. Besides, this framework is not currently available for public use.

Moreover, a Java-based context-aware recommendation library, named *CARSKit*, was proposed in [ZMB15, Zhe15a]. It contains context-aware recommendation algorithms, as well as some data structures and data processing algorithms. Specifically, the context-aware recommendation algorithms are divided into two types: transformation and adaptation algorithms. The transformation algorithms first pre-process the contextual data to obtain a 2D rating matrix (only with the information of users, items and ratings), and then apply a traditional recommendation algorithm. On the other hand, the adaptation algorithms directly incorporate contextual information into the prediction function. These context-aware recommendation algorithms can use traditional recommendation algorithms provided by the LibRec Java library [Guo14], included in CARSKit. Besides, it provides flexible configurations (of the algorithm and experimental parameters) and a standard platform for evaluations. Moreover, the library facilitates two methods to transform the categorical context information to binary format, and thus reduce the storage space and computational costs. However, CARSKit assumes that all the information needed to make the recommendation (users, items, contexts and ratings) is stored in a plain file. In addition, it does not support a distributed processing of the data. As future work, the authors proposed to include more state-of-the-art context-aware recommendation algorithms, as well as others that suggest appropriate contexts for users to consume the items [Zhe15b].

7.1.3 Examples of CARS

In the last years, context-aware recommendation systems have become one of the most popular topics of research in the field of recommendation systems. Moreover, several context-aware recommendation systems have been developed in various application fields. In this section, we present some examples of context-aware recommendation systems described in the literature for different domains. Specifically, we focus on domains such as the recommendation of songs, movies, TV programs, documents in digital libraries, news, and learning resources.

CARS for the Recommendation of Music

In the music domain, an example of context-aware music recommendation system (CAMRS) is the *Smart Radio* Web application, presented in [HC04b]. The main idea

of Smart Radio is to recommend music playlists, built by a listener, to other like-minded listeners. The recommendation model implemented in Smart Radio first uses a CF technique to obtain a candidate playlist to recommend, which is then refined with a content-based filtering, by using contextual data (e.g., artist and genre tags) associated with the playlist to indicate the user's current listening preference.

Another example is the C^2*_Music* system [LL07], that identifies the user's intention to listen music, as well the preferred type of music, by analyzing previous cases with similar context situations. The authors used a case-based reasoning approach, where the similarity of cases was extended to include the similarity of the contexts. The context data (e.g., season, month, day of the week, atmospheric condition, the lowest temperature, the highest temperature, and the average temperature of a day) were extracted from a weather bureau. For a user in a specific context, the recommendation model suggests the songs most listened by similar users under similar context cases. The similarity between users is based on the user's demographical data (e.g., taking into account the listening frequencies and the last dates when the songs were listened). This system was previously presented in [LL06], but in this previous work it recommends only the songs most listened by the same user in similar contextual conditions.

Another notable example is *uMender* [SYPT10], which is a context-aware recommendation system that combines musical content mining to select appropriate songs and context information filtering to group users with similar context conditions.

The emotional (or mood) state is a contextual factor that has been frequently used in CAMRS. An example of this is *MusicSense*, a contextual recommendation system (inspired by Google AdSense [Goo03]) that automatically suggests music while the user is reading web documents (e.g., Weblogs) [CZW+07]. In this scenario, the text of the web page is the context in which music is recommended. MusicSense matches the music and the content of a web page, in terms of the emotions expressed by the writers of documents and the composers of music songs. Both resources (music and web content) are represented as text documents, by using the "bag of words" model. The text information that describes the music is captured from its meta-data and online reviews. The authors proposed a generative model for determining the emotions of a text document. The songs most relevant to the context of a web page are then recommended.

Another example of mood-based CAMRS is described in [RjHH09, jHRJH10]. The authors built an ontology, called *COMUS*, to infer a user's mood from context information (e.g., the time, location, event, and demographic information), and then the inferred mood is matched with the mood transmitted by the songs, that is predicted from the music content. They used Emotion classifiers to map the feature vectors (e.g., pitch, rhythm, and tempo) of each piece of music into a single emotional state.

CARS for the Recommendation of Movies

In the movie domain, several context-aware recommendation systems have been implemented for the evaluation of contextual recommendation approaches. An example is the web application described in [ASST05], that was developed to test the

multi-dimensional recommendation approach proposed by the authors. The 62 users involved in the evaluation explicitly introduced into the system the applicable contextual information (e.g., the time –weekday, weekend, or do not remember–, place –movie theater, at home, or do not remember–, and companion -alone, with friends, with boyfriend/girlfriend, with family, or others–) and the rating (on a scale from one to 13) when watching a movie.

In [OTTK11], the authors also created an online application to obtain ratings entered by 89 users after watching a movie. In order to acquire the contextual information, the users filled a questionnaire in the application. The resulting information was used to generate the LDOS-CoMoDa dataset [Košl1, KOK+11] and evaluate the proposed contextual recommendation approach.

Similarly, in [CFTCD13], the authors needed to develop a web application to explicitly collect ratings (on a scale from one to five) and contextual information (e.g., time –morning, afternoon, night, or indifferent–, period of the week –working day, weekend, or indifferent–, and companion –alone, with my couple, with my family, with friends, or indifferent–) of movies watched by users (recruited via social networks). The information obtained was used to empirically evaluate contextual recommendation paradigms (e.g., pre-filtering, post-filtering and contextual modeling) proposed in the literature.

The challenge presented at the CAMRa 2010 workshop [BLSH10] focused on classification and ranking accuracy metrics of context-aware recommendation algorithms for movies, and it brought new ideas into the field of CARS. For the evaluation of the proposed contextual recommendation algorithms, two datasets (Moviepilot and Filtipset) were released.

CARS for the Recommendation of TV Programs

In the domain of television, a broad set of RS have been proposed in the literature [VPB+15]. In the specific field of CARS, one example is the web application *iFanzy*, that offers contextual recommendations of TV content to users [BSHA08]. For this purpose, the application uses a contextual semantic graph constructed from the available meta-data in several content sources (e.g., online TV guides, online movie databases such as IMDB, and broadcast). This graph is exploited to search available TV content and provide contextual recommendations. The implemented system supports explicit contextual information, such as the time, location and audience.

Another notable example of contextual TV program recommendation is the *PersonalTVware* system, presented in [dSAB12]. In this system, first the contextual user profile is created with the user's current context (e.g., location, day, type of the access device, and time), personal data (e.g., language, gender, age, and occupation) and preferences (e.g., director, actor, title of the TV program, etc.), which are explicitly entered by the user into the system. Then, PersonalTVware infers the contextual preferences for TV program genres, by using a machine learning technique (e.g., a decision tree classifier, Naïve Bayesian classifier, neural network, or case-based reasoning technique) and the current contextual user profile. Finally, the predicted genre class is incorporated into the contextual user profile, which is used to filter the TV

programs, by comparing the resulting contextual information with metadata of the TV programs. The resulting recommendation list can optionally be used to explicitly apply relevance feedback to the system, by selecting programs that are relevant for the current context. For each relevant TV program, the user explicitly specifies its genre. Therefore, thanks to this feedback, the contextual user profile is updated and re-learned by the system.

Moreover, the *smart TV* system was presented in [LKH14a]. It supports context-aware recommendations of multimedia content. The recommendation approach, included in the system, applies the contextual post-filtering paradigm. First, a social tag-based CF is applied from the user preferences, and then the resulting recommendations are re-ranked considering information about both the user's context (e.g., location type – public and private–, the crow around the user) and device context (e.g., multimedia product type and device infrastructure).

CARS for the Recommendation of Documents in Digital Libraries

The increase of documents (e.g., books, academic articles, journals, scientific papers, etc.) in Digital Libraries (DLs) has required the use of RS to facilitate the process of search and personalization of information. In the field of CARS, several studies have used context information to improve the recommendation of documents in digital libraries [NCI+11]. For example, in [JDA+11] the authors identified the contextual elements with more positive impact on the recommendation process for DLs, including user characteristics such as the purpose, activity, literacy, past searches, mental state, occupation, social status, expectations, and assumptions. In addition, in this study they had no evidence to say that the location and time had an impact on recommendations in scientific research ground. Moreover, the role of the context in recommendation systems was discussed in [CSS15a]. In this study, the authors conclude that the context information varies by domain, and so the context variables to consider should be selected for each specific domain.

From the years 2001 to 2013, a systematic review was conducted to identify the contextual information and context-aware recommendation approaches applied in the domain of academic digital libraries [CSS15b]. In this review, the authors provided a set of global conclusions. According to the context variables incorporated into a recommendation system, the authors classified the contextual information in categories such as users' context, document's context, and environment's context. Another conclusion was that the *citation of past studies* was the main contextual information exploited. Specifically, the recommendation system considers articles cited in a paper written by the user as relevant articles for her/him. Moreover, they detected that the traditional RS more used was the collaborative filtering.

CARS for the Recommendation of News

For the domain of news, different context-aware recommendation systems have been implemented. An example is the *News@hand* system presented in [CBC08a, CBC08b]. These systems use semantic technologies to recommend news. News contents, user

preferences, and context are represented as concepts in a set of domain ontologies. The exploited context information is obtained from the interaction (e.g., observed news) of the user during a session on the news website. An interesting feature of News@hand is that it supports recommendation of news for both a single user and a group of users.

Moreover, filtering the most appropriate news articles for individual users was addressed in [LCLS10, LCLW11]. For that, the authors modeled the problem of personalized news article recommendation as a contextual multi-armed bandit problem [LPP10], called contextual bandit. The proposed contextual-bandit approach learns to select news articles to users based on contextual information about the users and the stories.

Another solution is the application of context trees in the context-aware news recommendation system presented in [GDF13]. During the recommendation process, a partition tree organized in a hierarchy of contexts is used. For the authors, a context can be the sequence of articles read by a user, the sequence of topics, or the distribution of topics. In the tree, each node is a context and corresponds to a set of sequences or topic distributions within a partition. Then, a prediction model (called expert) is assigned to each context (or node) of the tree. The expert determines the probability of a candidate news item in that context, by considering the popularity of the news item, the freshness of the story, and the sequence of news items (or topics) that the user has seen so far. The user's browsing history is matched to the context tree and a path of contexts is identified. All experts associated with these nodes are responsible to generate the recommendation. Finally, the candidate items with the highest probability are recommended. Four versions of the proposed context-dependent news recommendation system and different traditional RS were included in a Personalized News RS framework (called *PEN recsys*), developed by the same authors [GF13] to facilitate the evaluation of several recommendation algorithms of news in an online environment [GFD+14].

CARS for the Recommendation of Learning Resources

In the domain of e-learning, a context-aware content recommendation system was described in [YNJ+07]. The authors used three ontologies to represent the knowledge about the learner (user's context), knowledge about the content, and the domain knowledge or learning field (e.g., computer science, mathematics, etc.). In the recommendation process, the semantic similarity between the user's context and the learning contents is first computed, and then a content list to recommend is generated. The resulting list can be interactively refined by the learner, taking into account different criteria (e.g., easier, more difficult, more interactive, less interactive, more generalized, and more specialized). When the learner selects one item of the list, the system provides a learning path (or study route) to guide the learning process. Finally, in addition to the recommendation, the system provides contents (e.g., examples, exercises, etc.) related to the course that the user is studying.

A survey of CARS for Technology Enhanced Learning (TEL) [BLJ+09] is presented in [VMO+12]. In this review, the authors identified different contextual di-

mensions used in the development of CARS for TEL. Besides, they analyzed exhaustively existing CARS in educational environments. Finally, they discussed a set of challenges related to the development and validation of CARS for learning.

7.2 Approaches for Context-Aware Mobile Recommendation Systems

With the continuous technological advances of mobile devices and communication technologies, the exploitation of contextual information in recommendation systems is being adapted to the domain of mobile computing, which implies considering context-aware mobile recommendations [LMCX13, PG13]. In the study carried out in [Ric10], Ricci discussed the goals of context-aware recommendations and their importance in mobile recommendation systems.

In this section, we first present several algorithms proposed in the literature of mobile CARS. These algorithms could be included in our architecture (see Section 3) in an easy way, thanks to its flexibility and genericity. Finally, we describe some examples of context-aware mobile recommendation systems developed for different domains. In Section 7.4, we mention some context-aware mobile recommendation works that use technologies mobile P2P.

7.2.1 Algorithms for Mobile CARS

In order to organize and facilitate the understanding of the works presented in this section, we classify context-aware mobile recommendation algorithms into two main categories: pull-based algorithms and push-based (proactive) algorithms. In the first case, we assume that the user actively (or explicitly) requests the recommendation. In the second case, the user, under certain contextual conditions implicitly receives recommendations, without explicit user requests. A common characteristic in both cases is that the contextual factors (e.g., location, temperature, transport way, etc.) are dynamic, in the sense that they can change over time.

Pull-Based Context-Aware Mobile Recommendation Algorithms

Pull-based context-aware mobile recommendation systems follow a request–response pattern. These systems only recommends items if a user makes an explicit (or query-based) request. A relevant aspect in mobile CARS is the dynamism of contextual information. In this line, several *pull-based context-aware recommendation* approaches have been proposed for mobile environments.

An example is the context-aware recommendation approach based on *interest resonance* (the common interest of users in some contextual attribute, such as location) for mobile environments presented in [CNL+12]. The proposed approach first obtains the interest resonance among users, by using a hash-based interest resonance mining algorithm, which relates the behavior of the users with contextual information obtained from the mobile devices. Then, the association degree between a user

and an item is combined with the user's rating on the item to build a context-aware recommendation index. The experimental results show that the proposed approach achieves a higher recommendation quality (in terms of both the MAE and RMSE) than state-of-the-art approaches.

In [UBSR16], the authors criticized the Context-Aware MF recommendation approach presented [BLR11] for the fact of modeling situations with explicit specific contexts to limit the dimensionality space. Hence, they decided to extend it and to propose the Latent Context Matrix Factorization Recommendation (LCMF) approach, which is a contextual modeling algorithm. The main idea of this new approach is to extract latent context data (e.g., regarding the location, ringer mode, speed, battery, activity, microphone, light, accelerometer, rotation, gyroscope, etc.) from a rich set of mobile sensors and use them to enrich the recommendation algorithm. In order to address the sparsity problem, the LCMF recommendation algorithm performs a selection of the best features. In terms of the RMSE, the proposed LCMF approach is superior to the Context-Aware MF model.

Push-Based Context-Aware Mobile Recommendation Algorithms

In mobile environments, where the user is moving and the context is highly dynamic, it is essential to provide precise recommendations and avoid overloading the user with the suggestion of many items. Generally, mobile devices such as smartphones have important limitations in comparison to traditional mobile or desktop computers; for example, they usually provide restricted input facilities (e.g., lack of a comfortable keyboard, small display sizes, etc.). So, a recommendation system can try to relieve the user from having to type or introduce significant information as an input, by using *push-based context-aware recommendations* rather than pull-based context-aware recommendations. A push-based context-aware recommendation approach automatically delivers recommendations to the mobile user in an appropriate context, without explicit requests from her/him.

For example, Woerndl et al. [WHBGV11] proposed a proactive contextual recommendation approach that pushes suggestions to the mobile user when the current situation (i.e., the context) is considered appropriate, without explicit user requests. The idea of this approach is to determine not only *which* items to recommend, but also *when* to make a recommendation. Hence, the proposed approach in a first phase periodically analyzes the current contextual conditions, and if the current context is appropriate, then a second phase is activated to examine the suitable items to suggest. For example, a gas station recommendation system can proactively suggest a gas station, when the remaining fuel level is nearly empty and a gas station is nearby, without much or any detour.

In the same scenario, another proactive context-aware recommendation approach was proposed in [LW12]. An important feature of this model is that it only considers the user activity as contextual information to determine when a recommendation must be generated. The user's activity (e.g., if the user is walking or not) is inferred from the contextual information (e.g., location, velocity, acceleration, distance to the previous point, etc.) obtained by sensors.

An agent-based architecture for context-aware recommendation was introduced by Neves et al. in [dMNCR14]. The *eAgora?* application is an implementation of the proposed architecture in a university scenario, which is characterized by an environment which is dynamic (mobile users), heterogeneous (several mobile devices, social interactions, and services available), intelligent (when the system is able to react to the environment), and contextualized (when the environment is context-aware). The application learns the user preferences in a continuous way, by using software agents, to adjust automatically the context changes and proactively recommend events.

From the perspective of artificial intelligent techniques (e.g., Bayesian Network, Fuzzy logic, and rule-based approaches), a contextual recommendation approach for mobile environments was proposed in [GK12]. This approach automatically recommends services (or actions related to the volume adjustment, call settings, profile, applications, etc.) in a specific contextual situation. For example, when in the library the recommendation model puts the device the vibrating mode automatically and also provides services like book search. The contextual information (e.g., day, time, location, temperature, etc.) is captured from sensors (e.g., accelerometer, temperature, humidity, etc.), and applications such as electronic calendars, address books, task lists, etc. In this contextual recommendation model, the context values captured from the sensors are represented as fuzzy values to define the context situations, the actions are executed under the current context conditions by using rules, and Bayesian Network techniques are used to classify the incoming calls (e.g., into high-priority call, low-priority call, and unknown call).

Moreover, the term *UbiCARS* has been proposed in [MP14] to emphasize the combination of the characteristics of both ubiquitous systems and CARS. Systems in this category are ubiquitous in the sense that they capture information from the environment and react to it. At the same time, they are context-aware because they consider the context in the recommendation process by using multidimensional contextual datasets. As an example, we could consider the case of recommending nearby items to a user while she/he is shopping (as in the use case presented in Section 3.1); a potential support system could be based on the use of NFC or other similar technologies to read RFID tags and identify products in the vicinity of the user, as well as to deliver product information that can be useful in that context.

7.2.2 Examples of Mobile CARS

In this section, we present several examples of context-aware recommendation systems for mobile environments in different domains (e.g., recommendation of mobile applications, restaurants, POIs, songs, learning resources, movies, services, etc.). As a summary, in Table 7.1, we provide an overview of the context variables and example application domains of existing work in the field of mobile CARS.

Mobile CARS for the Recommendation of POIs

In the field of mobile CARS, the recommendation of POIs has received the largest attention. An example of context-aware mobile application is COMPASS [SPK04],

References	Context Variables	Domain
[BBK10]	activity (travelling, working or shopping), calendar, conversations, activity streams of social networks	mobile applications
[CMBMRL12]	activity (applications used, application updated, etc.)	mobile applications
[KBCB12]	number of times the app was used, location, speed, time of the day, day of the week	mobile applications
[BNCM12]	location, trajectory, speed	POIs
[SPK04]	location, status, shopping list, schedule, weather, traffic information	POIs
[BNWP11]	location, time, price, gas level of the car	POIs
[jKAJ10]	location, time, interest, satisfaction level	POIs
[HG11]	age, first time in the zoo, companion with small children, time limit, annual ticket, weather, trajectories of POIs	POIs
[BRLW15]	travel time, visiting time, weather, time available, visit history	POIs
[DGMS16]	location, means of transport, weather, time of the day, day of the week	POIs
[MLCM13]	location, weather, time, social media sentiment	tourist attractions
[LWGD07]	location, time, weather, activity	restaurants
[GWH13]	location, time, activity (e.g., walking, hurry, etc.), companion	restaurants
[ZGS16]	location, transportation way, distance	restaurants
[WHBGV11]	location, time, companion, transport way, gas level of the car	restaurants and gas stations
[HPNM09]	week of the day, time, companion, weather	restaurant foods
[PYC06]	temperature, humidity, noise, illumination, weather, season, time, mood	music
[BKL+11]	driving style, road type, landscape, sleepiness, traffic conditions, mood, weather, time of the day	music
[WRW12]	activity (studying, running, walking, sleeping, working, shopping)	music
[SL15]	activity (stationary, walking, running, and driving), day of the week, time of the day, indoor/outdoor, place, temperature, humidity, pressure, sunshine	music
[BZ14]	background, preferred languages, schedule, activity, operating system, screen resolution, available memory, network bandwidth	learning resources
[LCVA12]	location, activity (sport, festival, landscape), personal information, date, time, information about the mobile device	photos
[RK17]	preview camera, time, weather, location	photos
[YZZ+06]	location and time	media
[ONM+12]	companion, time, location	movies
[SJM14]	topics of interests, hobbies, country, activity (e.g., doing sport, working, etc.), language, place, date, time, direction, movement, noise level, device platform, battery status, Internet connectivity	news
[RDB+08] [BBC+08]	week of the day, time of the day, weather, location	leisure activities
[LRHW15]	weather, time of the day, day of the week, temperature, companion, distance, crowedness, opening hours, item is in stock	shops

Table 7.1: Overview of example domains and context variables in existing work on mobile CARS.

which provides POI recommendations based on the location of a tourist and her/his interests. In [BNWP11] the authors also included the context in a recommendation system to recommend POIs such as gas stations, parking lots, or restaurants, in automotive scenarios (e.g., the driver of a car receives recommendations while driving). In [BNCM12] a context-aware system for mobile devices that incorporates implicit contextual information (the user's speed and trajectory) was proposed. The authors defined an area of interest (AOI) such that only the POIs (e.g., restaurants, museums, etc.) within that area are recommended to the user.

Moreover, the classical collaborative filtering approach was modified in [jKAJ10] to include several context dimensions (e.g., location, time, interest, and satisfaction level) by applying data mining techniques (e.g., decision tree-based classification rules to understand the needs of the users). The proposed algorithm was evaluated in the context of the recommendation of places for shopping, eating, enjoying, drinking, and learning. The inclusion of Context-aware Collaborative Filtering (CaCF) into mobile guides was investigated in [HG11]. The authors started from the idea that similar users usually choose similar POIs in similar contexts. They included contextual information into the CF through contextual pre-filtering and contextual modeling. The methods proposed were evaluated with available GPS trajectories collected from the Vienna Zoo (Austria), with the purpose of offering relevant context-aware POI recommendations to the visitors.

In [BLPR12], the authors proposed a methodology that supports the development cycle for CARS. To put it in practice, they developed a context-aware mobile recommendation system prototype called *ReRex*, that suggests interesting POIs for tourists according to the value of several contextual conditions.

According to [EBRT13], a better user profile will lead to better recommendations. This can be obtained by applying active learning techniques to ask the user to rate specific selected items (e.g., popular and non-redundant items) for which the knowledge about the user ratings is expected to improve the quality of her/his profile. To evaluate the approach proposed, a mobile CARS to recommend POIs for tourists in the Alto-Adige region in Italy was presented in [EBRT13].

The work in [VHR12, GH12] presents a model to generate context-aware mobile recommendation systems that recommend places (e.g., restaurants, stores, cinemas, supermarkets, etc.) where bank clients have paid with their credit cards. The authors used banking data to obtain information based on real person's actions and banking history.

Recently, a context-aware recommendation system that suggests popular itineraries of POIs was described in [DGMS16]. During the recommendation process, it considers the number of check-ins on social networking services (e.g., Foursquare), the user's profile, and the contextual information (e.g., current location, means of transport, weather conditions, time of the day, and day of the week). The location and means of transport are captured from GPS and accelerometer sensors, respectively. Moreover, the weather conditions are acquired by querying weather services, by using the current location. The authors applied a directed graph to determine the shortest routes to recommend, where the nodes are the POIs and the edges contain

the traveling time from one node to another as a weight, by considering the user's means of transport. Previously, the POIs are filtered based on the current context information.

As a final example, a context-aware probabilistic matrix factorization approach for POI recommendation was proposed in [RSHS17]. It uses textual information, geographical information, social information, categorical information, and popularity information, to generate a POI preference score, which is then integrated into a probabilistic matrix factorization model.

Mobile CARS for the Tourism Domain

The tourism domain is the application area that has received the largest attention in the recommendation of POIs. For example, in [BERS13, BER14], the authors described a context-aware recommendation system for tourists, named STS (South Tyrol Suggests), that takes into account the impact of the weather conditions at a specific POI. In a similar direction, [MLCM13] proposed a hybrid mobile recommendation system that exploits contextual information to support tourists in making decisions about the attractions to visit.

In [BRLW15], a proactive mobile recommendation system pushes information about POIs to tourists when the current contextual conditions (e.g., travel time, visiting time, weather, time available to the user, and the user's POI visit history) seem appropriate. The proposed approach considers user preferences as well as several heuristics to predict the overall utility of pushing a POI recommendation to the mobile user in the current contextual situation.

Recently, the *POST-VIA 360* system was presented in [CPGPSM16]. It is a social and context-aware recommendation system that offers POIs to tourists based on data from previous visits and the social environment of the tourist. In the tourism domain, other context-aware mobile recommendation systems have been proposed in the literature (e.g., [GLX+11, Liu14]).

Mobile CARS for the Recommendation of Services

In addition to POI recommendation systems, numerous service recommendation applications have been proposed in the tourism and mobile applications. An example is the context-aware RS presented in [TZAQL12], that proactively pushes recommendations of suitable services (e.g., hotels, restaurants, events, etc.) to tourists, by using multi-agent technology. In [XDL+16], a novel approach to recommend services or applications on mobile devices to users has been proposed. The authors presented a user behavior model by considering the user's mobile context information (e.g., time and location) to describe the user state. Based on the user behavior model, they built a model to explain how the sequential service invocations are generated according to the changes of user states. Finally, they trained a logistic model tree classifier (with past context information and user states) to detect the user's current state given a specific mobile context, and thus to recommend services to the mobile user according to her/his state.

Moreover, several context-aware service recommendation systems based on role mining [FBB10] have been proposed in the literature [WZH+12, WCH14, YWC15]. These works are motivated by the idea that users (or a user group) with the same role are likely to share the same interests and behavior patterns. The user group represents the abstract characterization of user behaviors within a certain context. Users can play different roles, and their roles change dynamically as the context changes. Once a user's role in a certain context is automatically detected, the services closely related to that role can be recommended to the mobile user.

Recently, in [NACH17], the authors presented the design and implementation of a context-aware service recommendation system for mobile users in a smart city. The proposed system is able to suggest a set of services to a mobile user according to her/his context (e.g., the location, city, destination, speed, time, date, season, temperature, and weather) in real-time. The recommended services can be explicitly requested by the user or automatically pushed by the framework to identified users.

Although they are outside the field of recommendation systems, there are a couple of additional proposals worth mentioning. On the one hand, the SHERLOCK system [YMII14] offers a framework for LBSs, which exploits ontologies and semantic techniques to share knowledge among devices and guides the user in the selection of the service that best fits her/his needs in the given context. On the other hand, the Long-Life Application (LLA) approach [KIRD17] proposes the use of a single long-life application for all purposes, that adapts itself dynamically to the current situation.

Mobile CARS for the Recommendation of Restaurants

For the domain of restaurants, a proactive recommendation system that pushes recommendations to the user when the current context is considered appropriate, without explicit user requests, was described in [WHBGV11]. For example, a context-aware restaurant recommendation guide suggests a restaurant to the mobile user, when she/he is walking near the restaurant of her/his preference and it is lunch time. Along these lines, [GWH13] evaluated the impact of proactivity in the user experience: whether users would accept proactive recommendations, how to present the recommended items, and how to properly notify the users. To answer these questions, the authors evaluated two mobile user interfaces (a widget-based interface and a notification-based interface) for context-aware restaurant recommendation, based on the proactive recommendation model proposed in [WHBGV11]. Experimental results showed that the widget-based interface was preferred by the users. In [BA12], the authors presented the situation-aware proactive recommendation system SAPRS, that pushes information about relevant restaurants to the user at the right contextual situation only. In the first phase, SAPRS applies fuzzy logic as an inference technique to handle the uncertainty of the current situation, and if the situation is appropriate then the CF recommendation model is activated in a second phase. The same authors incorporated explanations to the recommendations provided by SAPRS, in [BASJ14]. Similarly, in [ABF16] a proactive context-aware recommendation system that considers only the location and the time as contextual information was presented.

Moreover, several pull-based mobile CARS have been described in the literature

for the domain of restaurants. An example is the work presented in [LWGD07]. The authors presented a CF-based framework, based on OLAP (Online Analytical Processing) and multi-dimensional CF, that provides contextual recommendations; the framework was implemented in an existing m-commerce platform of the food industry. Another novel multidimensional approach for context-aware recommendation in m-commerce was presented in [HPNM09] and evaluated in the context of a recommendation system for restaurant meals. Recently, a pull-based context-aware recommendation system for mobile environments called *Co-ARS* was developed in [ZGS16]. The system recommends restaurants by considering contextual information, such as the user's location, restaurant's location, user's preferred transportation mode, network distance between the user's current location and the destination, and overall rating of the restaurant. The user's location was automatically acquired from the embedded GPS sensor in a smartphone. The transportation mode (e.g., stationary, walking, biking, or driving) was detected by using a Bayesian Network classifier [SES15]. During the rating prediction process, the proposed recommendation model favors nearby restaurants, by considering the distance between the user's current location and the restaurant's location and the mode of transportation. Besides recommending a list of nearby restaurants, for each of them the optimal route and travel mode is suggested, by using the Google Maps app.

Mobile CARS for the Recommendation of Music

In the domain of context-aware music recommendation, the *CA-MRS* system was developed in [PYC06]. It uses a fuzzy system to pre-process information (that is represented as a fuzzy membership vector) obtained from different sources (e.g., sensors, the user profile, and the Internet), Fuzzy Bayesian networks to infer the current context (e.g., temperature, humidity, noise, illumination, weather, season, and time) from the fuzzy membership vector, and the utility theory to obtain the final score of music based on the user preferences (e.g., genre, age, tempo, and mood) and context. The user preferences are manually entered by the user.

InCarMusic is another example of context-aware mobile recommendation system that offers music recommendations to the passengers of a car [BKL+11]. For the recommendation of songs for daily activities (e.g., studying, running, walking, sleeping, working, and shopping), a novel probabilistic model was proposed in [WRW12], which integrates contextual information (collected through mobile devices) with music content analysis.

A general framework for the recommendation of social events (e.g., music events) was proposed in [BHD13, BHHD13]. It supports the implementation of context-aware recommendation engines for mobile platforms. The main idea of the framework is based on the hybridization of traditional CF with contextual information. Moreover, it uses a MapReduce programming model [DG08] for the distributed data aggregation and blending of multiple context-dimensions.

Another example is the music recommendation system presented in [SL15], that suggests songs based on contextual sensor information (e.g., activity, day of the week, time of the day, indoor/outdoor environment, place, and atmospheric conditions).

Raw sensor data are fused and passed through a cascade of Fuzzy Logic models to infer the user's context, which is then used to obtain a candidate music list from an online music streaming service (SoundCloud). Finally, the candidate list is filtered, by considering the genre preferences that the user dislikes.

Recently, a multimedia recommendation framework, called *RecAm*, was proposed in [ARD+16]. It incorporates contextual information (e.g., the time, health conditions, emotions, calendar, location, etc.) into the recommendation process. The purpose of this framework is to facilitate the recommendation of multimedia content by identifying the user's context through adaptive user interfaces in ambient intelligent (AmI). For example, a prototype that uses the proposed framework would be able to detect the stress level of a person (by capturing the heart signal) and recommend suitable songs that can decrease the stress level. Besides, the prototype could modify the environment of the room (e.g., adjusting the volume of the music and the level of light) according to the preferences and current context of the user, by using AmI interfaces.

Mobile CARS for the Recommendation of Movies

In the movie recommendation domain, *Cinemappy* was described in [ONM+12]. It is a mobile application that uses two different contextual filtering paradigms and a content-based recommendation approach to suggest movies and movie theaters to mobile users. The recommendation engine first applies a pre-filtering approach, by using contextual attributes such as companion and time. Then, a content-based recommendation model is used to match movies with contextual user-profiles, by considering the DBpedia [ABK+07] as the only information source. Finally, a post-filtering approach is used to obtain the resulting movies or cinemas, considering geographic information. In the movie showtimes domain, another example is the *RecomMetz* application, proposed in [CMVGRG+15]. In this work, the authors put forward a recommendation system for mobile devices based on the use of semantic web technologies in order to provide users with suggestions about the best movies to watch and the most appropriate theaters to go to. To calculate the recommendations, they use context data such as the time, the user's location, or crowd measures.

Based on a dimensional recommendation model and a hybrid processing approach, the *CoMeR* platform [YZZ+06] supports context-aware media recommendation for smartphones. To validate the approach, a context-aware movie recommendation system for smartphones, called *ContAwareMovie*, was developed by using CoMeR. In [LT16], the authors incorporated contextual information (e.g., the location, companion, mood, etc.) in two collaborative filtering approaches (memory-based and model-based recommendation) for multimedia recommendation in a mobile environment. In order to evaluate these approaches, the authors implemented a system for movie recommendations. The experimental results showed that the proposed approaches outperform traditional methods.

Mobile CARS for the Recommendation of Learning Resources

In the context of learning, Wan and Wu [WW11] applied context-aware technologies, recommendation algorithms, and RFID technology to implement a ubiquitous learning system. It is able to automatically detect learning resources in the user's current context for the recommendation of learning activities. For the same domain, the work in [Asa13a] presented a framework for the recommendation of learning resources for learners in a mobile social learning community.

Another case is the context-aware recommendation system proposed in [BZ14]. It provides learning services adapted to the learner's context (e.g., learner's background, preferred languages, and learner's schedule), activity context (e.g., accessed services, consumed learning resources, adopted learning sequences, etc.), device context (e.g., operating system used, screen resolution, and available memory) and environment context (e.g., temporal and spatial contextual information, network bandwidth, etc.). These contexts are represented by using different ontologies. When the context changes, the system is able to identify the new contextual features and translates them into new adaptation constraints in the operating environment, thus enabling context-aware learning service recommendation.

Recently, an ambient intelligence context-aware affective recommendation platform, called *AICARP*, was developed in [SSBRS16]. The main goal of this platform is to deliver more interactive educationally-oriented recommendations. For example, the system is able to detect if the learner is getting nervous while studying for an oral test. In this situation, the system interacts with her/him without interrupting the learning task and suggests her/him to reduce her/his breathing speed to calm her/him down. The authors used an Arduino-based infrastructure to sense the changes in the learners' affective state, and thus be able to interactively push recommendations.

Mobile CARS for the Recommendation of Applications

Due to the popularity of smartphones and the growing number of mobile app made available, a number of research works on context-aware mobile app recommendation have appeared in the literature. For example, in [WSW07], the goal is to recommend mobile applications (e.g., a mobile restaurant guide) to users that have a context similar to the one existing for other users when they installed the applications. The authors of that work proposed to hybridize traditional recommendation systems to manage contextual information (e.g., the current location of the user).

In other works, the application recommendation approaches are focused on the use of mobile applications as a relevance measure related to different contexts. An example is the *appazaar* prototype [BBK10], which suggests mobile applications to mobile device users based on the actual usage of the applications in different contexts. Its authors describe different dimensions and techniques to obtain information about the users, items, ratings, and contexts. Another example is *Which App?* [CMBMRL12], an application that recommends mobile applications to users by considering the user's activity (e.g., which applications are used, how many times the application has been used, how many times it has been updated, etc.), previous tagging of applications,

her/his preferences, and similarities with other users. The user's activity is collected continuously and with a certain frequency. The *Djinn* model [KBCB12] is a context-aware recommendation algorithm using tensor factorization, by considering implicit contextual information such as the number of times the app was used, location, user's speed, time of the day, and day of the week. The proposed model was validated with data collected by the *appazaar* application. Finally, a recent example is the Android widget implemented in [MAP16], that recommends mobile applications, calls and SMSs to the user at a given day of the week and time of the day, depending of her/his activity pattern.

Mobile CARS for Recommendations in Shopping Scenarios

In order to facilitate the consumer shopping process, several mobile CARS have been implemented. For example, in [CHPY15] the authors implemented an Intelligent Shopping-aid Sensing System, called iS^3, for online shopping. It is a context-aware recommendation system that uses clustering techniques and association rule methods to provide product recommendations to customers according to their historical purchasing records.

Another example is the context-aware mobile shopping recommendation system presented in [LRHW15]. In this system, the authors introduced the contextual information (e.g., the weather, time of the day, day of the week, temperature, user's company, distance to shop, crowdedness, shop opening hours, and if the item is in stock) into the recommendation model by applying the pre-filtering and post-filtering paradigms. First, a pre-filtering approach is used to determine which items of the knowledge base are relevant to the user, by considering contextual factors of the shop. Then, a post-filtering step is applied to filter the items appropriate to the current user context, by using a nearest neighbor algorithm.

In [SGM15], the authors studied a set of context factors used by others researchers in the literature for e-commerce RS. They considered that the user's real-time state of mind and their current budget are important factors for commerce. Finally, another work to be highlighted is [LLH17]. It intends to build routes for users in a shopping mall by taking into account contextual information of both retailers and visitors. An interesting property of this work is the consideration of the friends' opinions as part of the recommendation process.

Mobile CARS for the Recommendation of Other Items (Photos, News, Activities, etc.)

In the domain of photos, *MMedia2U* [LCVA12] is an example of mobile photo recommendation system which exploits the user's contextual information as well as the existing context when the photo was created. Another example is *ClickSmart* [RK17], a recommendation system that suggests real-time viewpoints for photography assistance in popular tourist locations, based on the preview of the user's camera, current time, and user's geolocation. Besides, the authors studied the impact of the context (e.g., time and weather) for the viewpoint recommendation.

Different recommendation systems have also been developed in the scenario of news. An example is the recommendation system of news proposed in [SJM14], that uses the current context of mobile users and the format in which the news items should be represented. In [PCV+16], the authors studied the influence of the contextual conditions in the recommendation of mobile news services.

Several existing works in the literature of mobile CARS are focused on the user's activity to generate recommendations. For example, *Magitti* is a scalable architecture for context-aware activity-detecting mobile recommendation systems [RDB+08, BBC+08]. It infers leisure time activities based on the context and patterns of user behavior. [DTB12] presents an activity-aware recommendation system for teams of medical professional working in hospital operating rooms. It suggests relevant virtual actions (e.g., retrieval of information resources and initiation of communication with professionals outside the operating rooms) based on the current state of the operation (detected from sensor data) and in similar past situations, by using machine learning techniques.

7.3 Approaches for Location-Aware Recommendation Systems

An important subset of CARS is represented by the so-called *Location-Aware Recommendation Systems* (*LARS*). These systems only consider the dimension *location* in the multidimensional context. In this section, we first present some approaches described in the literature for LARS. Then, we mention relevant examples of LARS in several application domains.

7.3.1 Algorithms for LARS

In recent years, thanks to advances of mobile devices, ubiquitous computing, and wireless communication technologies, a significant number of works have been carried out in the field of LARS [dCRHITLH15]. An example is the system presented in [LSEM12, SLEM14], which exploits location-based ratings to provide recommendations. To obtain spatial ratings, the authors applied an approach of user partitioning based on the user locality, the scalability to large numbers of users, and the influence of the users, to control the size of the neighborhood. For spatial items, a travel penalty was applied (favoring the closest items). The collecting process of the spatial ratings was motivated by a study carried out on the MovieLens dataset [Gro16], that associates the locations with the user's ZIP codes (i.e., spatial ratings), and the Foursquare dataset [Fou13, YZYW13], which contains information about places visited by users (i.e., spatial ratings for spatial items). Recently, and along the same vein, the authors of [AD15] presented *LARS**, that also recommends items based on location-based ratings, by using user partitioning and travel penalty techniques. In this case, the location is obtained from the IP address of the user's mobile device.

A similar goal was pursued in [YCC+15], where the authors presented *LA-LDA*,

a location-aware probabilistic generative model that uses location-based ratings to model user profiles to produce recommendations (e.g., suggestions about restaurants) as well as to mitigate the well-known cold start problem. They considered the three types of location-based ratings proposed in [LSEM12] (i.e., spatial user ratings for non-spatial items, non-spatial user ratings for spatial items, and spatial user ratings for spatial items). In [KCL09], the authors proposed a *location-based service recommendation model (LBSRM)* that combines relevant elements of LBS and recommendation technologies. Firstly, the model filters information based on the user's location, and then it recommends relevant mobile information services by using clustering techniques. With a similar spirit, the authors of [DGY+14] recently integrated LBSs with recommendation techniques to present a hybrid recommendation model.

Other approaches consider the impact of the locations not only as a pre-filtering step but directly in the application of collaborative filtering. For example, [DMG12] uses Voronoi diagrams to decompose the user's space and then it uses them in a spatially-aware collaborative filtering algorithm; specifically, the authors explored the concept of spatial autocorrelation to cluster similar values on a map, by using statistical measures. In this approach, the ZIP code of the area is used to identify the user's location. A location-aware collaborative filtering was also proposed in [SU11], which uses the user's location to recommend web content in real-time, increasing the diversity of recommendations; specifically, the authors determine the diversity using the Levenshtein edit distance [YB07] between attributes of items (e.g., locations, tags, titles and URLs) to try to address the handicap of popularity bias without affecting the performance. Moreover, recently, the authors of [YHL+15] proposed a location-sensitive recommendation approach in ad-hoc social network environments.

With the development of the Web 2.0, some works focus on the combination of mobile technologies with traditional social networks, giving rise to *Location-Based Social Networks (LBSN)* [BZWM15], such as Foursquare [CS09], Facebook Places [ZSHM10], and others. The emergence of this new kind of social networks allows connecting with friends, share locations (and/or photos, videos, etc.), receive recommendations of places (e.g., restaurants), etc. The main research topic covered is how to effectively combine the information provided by social networks to offer more accurate recommendations. For example, a user could trust particularly the recommendations offered by her/his friends, but not all the user's connections are necessarily real friends. Analyzing in depth how information about the user's social interactions in real-time (e.g., a tweet or photo published by the user, or a conversation with a friend) could be exploited in the context of LARS is an issue that has not been explored in depth so far.

A Markov-based technique presented in [AJJR14] improves the quality of location-aware recommendation systems by using the location information of items. In the Markov model, the authors consider each item as a state. The states are defined as the history of items viewed (or visited) by the users, and the transition probability is calculated according to the preferences (likes) of items by the users in the past. In general, the recommendation approach suggests the items with the highest likelihood estimation, by taking into account the location (i.e., a greater geographical distance

among the items decreases the probability estimation). In [PZC+14], a collaborative filtering recommendation approach was presented, focusing on the specific case of suggesting geospatial locations (e.g., latitude and longitude) where mobile users can take photos. The final list of locations to recommend must be within a (user-defined) suitable distance from the physical position of the user. Instead of exploiting the users' locations, the authors used three million geotagged photos taken from smartphones (i.e., photos implicitly containing geocoordinates). In [SK11], data mining techniques (e.g., clustering models) were used to recommend items to the mobile users by considering the user's location. In [QZC+14, QZH+14], the authors presented an improvement of collaborative filtering that combines the user's geographical information and the content of items in order to learn location-based user group preferences, considered by the authors as a rating distribution of a group of items. According to the study performed with the MovieLens dataset, the user group preference has strong correlation with the location of the user. In [MLM+17], the authors proposed a framework for LARS, called *APPLET*. The main motivation of APPLET is to protect the user privacy information (e.g., locations and recommendation results) in a cloud environment. For this purpose, they applied several encryption techniques in an item-based collaborative filtering model.

7.3.2 Examples of LARS

In this section, we present some relevant examples of location-aware recommendation systems for several application domains. Specifically, we focus on domains such as the recommendation of POIs, restaurants, news, shops, songs, educational materials, and TV shows.

LARS for the Recommendation of POIs

One of the most common application domains of LARS is suggesting interesting points around the user. For instance, a collaborative location-aware filtering approach to recommend POIs to mobile users, which exploits the location as a relevant element for the recommendation of items (e.g., restaurants) near the user's current location was proposed in [HNV06]. The approach proposed is the result of combining user-based collaborative filtering techniques with a location-based partitioning method (i.e., it allows an adequate rating database partitioning based on the location), with the goal of achieving a high scalability. That work validates the hypothesis that users who live nearby tend to visit the same local places. The proposal in [CFP+12] attempts to solve the problem of location-based context-aware recommendations of POIs by using a multiagent system architecture [Wei13]; the use of agents facilitates the collection of information about POIs' available on the Web. Another example is the location-dependent collaborative filtering system presented in [THK11], that analyzes the mobile user's moving features (e.g., moving direction, position, and speed, obtained through a GPS receiver) and the POIs, in order to recommend to the mobile user those items of interest that are in a region near the user's current position and in the same direction.

A ubiquitous location-based recommendation algorithm that suggests relevant places to mobile users is presented in [SBCH12]. The system, named "I'm feeling LoCo", considers the user's profile and the places near her/him during the recommendation process. It automatically infers the user's preferences (by mining social network profiles) and considers spatio-temporal constraints in the recommendation process. The physical constraints are delimited by the user's location and the transportation way (e.g., driving a car, riding a bicycle, or walking).

A location-based and preference-aware recommendation system that suggests venues (e.g., restaurants and shopping malls) within a geospatial range was presented in [BZM12]. It learns the user preferences automatically from the user's location history and infers the user's expertise (e.g., in categories such as Chinese food and shopping mall) in several cities. During the recommendation process, the system filters the candidate local experts in a geospatial range (defined by the user) and suggests the venues that match the user's preferences and the social opinions of the selected local experts. This type of system has the advantage of providing venues not only near the area where users live, but also in cities unknown to them. A similar goal was pursued in the *Location-Content-Aware Recommendation System (LCARS)* proposed in [YCS+14], which recommends venues (e.g., restaurants) or events (e.g., concerts and exhibitions) within the city where the query initiator is located, by using the probability of influence of the personal interests and local preferences of the users. One of the main goals is to alleviate the data sparsity problem (the new city problem) based on the location and content information of spatial items. As a final example, the *PECITS* system [TR09] that provides location-aware recommendations of POI paths (e.g., a list of several connections that the user could take to reach a certain POI, by using public transportation and by foot) in the city of Bolzano (Italy).

LARS for the Recommendation of Restaurants

Various location-aware recommendation systems have been developed in the restaurant domain (e.g., Yelp [Yel04], Tabelog [Kak04], etc.). In [YH12], a recommendation system that suggests restaurants close to the user's location is presented. An interesting contribution of this work is that users can visualize information (e.g., rating, popularity, and distance) of the recommended restaurants as $3D$ bar graphs, applying augmented reality.

Another example is the location-based recommendation architecture for dynamic and ubiquitous environments proposed in [GS13]. The authors combine, in the proposed architecture, the ideas of location, personalization, and content-based recommendation. Finally, an Android application that recommends restaurants based on the user's location was recently introduced in [HKBO16].

LARS for the Tourism Domain

In the tourism domain, the recommendation process implies suggesting a set of products or services that support traveling and tour planning (e.g., attractions, accommodations, restaurants, and activities). For example, the authors of [LMZW10,

NBSM12] integrated tourism mobile commerce and location-aware features into a traditional recommendation system to provide real-time recommendations for visitors, by taking into account the locations and the ratings of the attractions. Similarly, an architecture for location-based recommendation was proposed in [YpC09], which supports personalized tour planning for mobile tourism applications by using rule-based recommendation techniques.

Along the same line, the authors of [CSdVR11] presented a system that recommends touristic places based on the user's visiting history in different regions (e.g., cities or countries). To recommend locations, a set of geotags representing the latitude and longitude where a user took a photo (manually set on a map or automatically obtained from the GPS device) is exploited. This is considered useful to plan a touristic visit to a new city or country.

LARS for Recommendations in Shopping Scenarios

In the field of mobile commerce (m-commerce), several types of LARS have been designed and presented in the literature to suggest a variety of products and services that may be of interest to users. An example is the location-aware recommendation system presented in [YCD08], that recommends vendors' web pages to interested customers in mobile shopping. Another example is *CityVoyager* [TS06a], a recommendation system based on the user's location history, which is obtained by using a GPS device. It recommends shops to the users based on the locations of previous shops visited.

In order to avoid the need to type text, along with the associated spelling problems and possible ambiguity, when the user needs to specify the types of items she/he is interested in, an interesting proposal was presented in [ZOON12]. Specifically, the location-based shopping recommendation system proposed uses an image of the desired item (e.g., shoes, clothes) provided by the user, as the query, as well as the smartphone's GPS coordinates, to recommend retail shops (with information including their GPS coordinates, promotions, and special offers) to mobile users.

LARS for the Recommendation of News

Most LARS use the user preferences and the distance between the current user's location and the positions of the items for the recommendation of relevant items. However, it is not usual to enrich the previous approach by using existing relations between items and tagged locations (e.g., geographical metadata of news articles), which could have an impact on the recommendations.

Thus, the authors of [AW14] proposed an interesting spatial model for location-based serendipitous recommendation of news articles. For that purpose, they studied the existing associations between the user's current location and the location data available in geographical metadata of news articles. The introduction of serendipity in traditional collaborative filtering implies modifying the recommendation approach to emphasize the novelty (or the surprise) and discover useful items for the user, in exchange of some accuracy.

A location-based social networking system for mobile devices, named *Sindbad*, was proposed also in the field of news [SBE+12]. With Sindbad, the user can receive her/his friends' news based on their locations, as well as messages posted by them. Moreover, its recommendation system also suggests spatial items (e.g., restaurants) and non-spatial items (e.g., movies) based on the users' locations, the items' locations, and the ratings provided by friends. For that purpose, the location-aware recommendation module LARS proposed in [LSEM12] was used.

LARS for Other Scenarios (Music, Health, Learning, etc.)

Finally, it should be highlighted that, although the domains examined previously are the most common ones, there are other possible use cases. For example, in the area of music, the authors of [BKR13, KRS13a] tackled the problem of providing location-dependent music recommendations by using emotional tags (e.g., melancholic, cold, heavy, animated, etc.) related to the music and the places of interest. With this idea, they developed a mobile location-aware recommendation system named *PlayingGuide*, that suggests and plays appropriate music for a place of interest for the user (e.g., the user might hear a specific music while visiting a place of interest in a city). As another example [CS14], the authors implemented a recommendation system, called *Just-for-Me*, that suggests songs based on the user's location, global music popularity trends (detected from Twitter streaming data), and music contents. The system unifies these aspects through of a probabilistic generative model built by the authors.

Another interesting work is *Motivate* [LJdVT11], which presents a context-aware mobile recommendation system that promotes a healthy lifestyle. It recommends different kinds of useful advices to the user (e.g., take a break, walk/cycle to a park, go to a museum), by considering the location of the user, the activities in the user's agenda (e.g., go to work, work, have lunch, go home, have dinner, and busy), the time (e.g., the start and end time of an activity), and the weather (e.g., bad, fair, and good) as context parameters.

There exist also some attempts to use the location for recommendation in e-learning environments. The approach in [EBOY07] recommends educational materials and peer learners who are nearby, by using RFID to detect the learner's environmental objects and her/his location. The system also allows the learners to share knowledge, interact, collaborate, exchange individual experiences, and visualize the objects that surround the learner, the space of learning resources, and the distance to possible peer helpers.

Recently, in [WLX16] the authors presented a location-aware recommendation system that suggests popular TV shows, by considering the social network information of users from different locations, as well as the audio and visual features of TV shows. These aspects are used as a regularizer into the collaborative filtering model to further refine the recommendation.

7.4 Approaches for Mobile P2P Recommendation Systems

Most recommendation systems are implemented in a centralized way. However, some researchers have also considered a P2P architecture as an interesting alternative to reduce the calculation complexity of centralized recommendation algorithms. In this section, we present some mobile P2P recommendation approaches described in the literature.

One of the first approaches that addressed the problem of scalability in recommendation systems was presented in [Tve01]. The authors proposed a P2P-based collaborative filtering approach that suggests products and services to mobile customers. Recommendation queries are broadcast to neighbor peers, which are software assistant agents interacting with a mobile customer. In the same application domain, Zhang et al. [ZLL11] proposed a distributed recommendation approach based on cloud computing. In this distributed architecture, the authors divide the rating database into sub-databases (by using the K-means clustering method), which are stored in distributed cloud servers. The recommendation queries are submitted to the different cloud servers in a collaborative way and the data are updated by using a Distributed Hash Table (DHT).

Another early example where the authors propose a P2P architecture to achieve a reasonable storage structure for opinions about items is presented in [HXYS04]. In this proposal, the users maintain information as simple containers; specifically, DHTs are used and new entries location are calculated by measuring similarities.

Another work worth mentioning is [PM06], where the authors present a P2P recommendation system that exploits trust as a cornerstone for users to share their information with others. In this context, the trust between two users is interpreted as their capability to predict the ratings of common choices.

In [KP10], the authors presented a P2P recommendation approach different from the one that we propose in Section 3.2.3. It is based on a naïve Bayesian classifier and uses a P2P topology that preserves the privacy of users. Through a P2P network, the users can communicate with each other and exchange data to generate predictions. For example, when a specific peer needs to predict the rating of an item, it sends a request of ratings about that specific item to other peers in the network. The peers that rated that item predict the probability values for each class (likes or dislikes) and then send the calculated values to the peer submitting the query. Besides, in order to preserve the privacy, the authors apply a specific mechanism to hide rated items in each peer. A challenge for the practical application of this approach in a mobile P2P network is how to ensure the routing of the answers to the originating peer by using only wireless ad hoc communications, which is difficult due to the continuous movement of the mobile peers.

With the same flavor, an interesting P2P recommendation system for large-scale data sharing was presented in [DPK11]. This work puts forward a mechanism that is able to communicate information to other peers based on a semantics-based gossip protocol. Moreover, the authors also propose a query routing algorithm able to

identify relevant peers, in order to avoid the problem of information flooding, by calculating similarities among peers and topics.

Another relevant example is the *iTravel* system, that recommends attractions to tourists in a mobile environment [YH13]. iTravel supports mobile P2P interactions, using short-range wireless communications such as Wi-Fi or Bluetooth, to exchange the ratings of the visited attractions among the mobile devices of the users. The authors presented three data exchange methods. In the first method (*unconditional data exchange*), when the recommendation application of a user detects another iTravel application nearby, it sends to that iTravel application all the ratings stored in its database; to avoid the information flooding problem, the maximum number of propagations of ratings is limited. The second method (*preference-based data exchange*) avoids the information flooding problem by considering the user's preferences when deciding which information to exchange: iTravel propagates only the rating lists of similar users (i.e., rating lists that are similar to the recipient's rating list). This alternative may be inappropriate for the case where the user is surrounded by other users with tastes different from their own. Hence, the third proposed method (*hybrid data exchange*) is a combination of the two previous ones.

Finally, a context-aware mobile recommendation system (called *PPNews*) based on Bayesian Networks and P2P technologies was presented in [YYN10, YYN12]. It proactively pushes news articles to users in a mobile P2P network, by considering the user's context and the content of news. The authors implemented it by using *JHPeer*, which is a hybrid P2P framework built by themselves, that supports peer-to-peer communications. JHPeer provides basic context-aware services to handle context queries or deliver contextual information to peers. During the recommendation process, first the mobile device establishes communication with one of the peer servers to communicate the user's profile information. Then, the context service (JHPeer) installed on the user's mobile device automatically delivers context information (e.g., location and usage pattern) to the server. Finally, the server proactively recommends the top-k news articles to the mobile user based on the user's profile, the context information, the news content, and the ratings stored by the peer. The rating prediction is determined by using a hybrid recommendation approach (content-based filtering and collaborative filtering).

7.5 Approaches for Synthetic Data Generation

In this section, we analyze some existing approaches in the literature to build synthetic datasets. Most of them are aimed at providing collections of data for fields in which it is rare to find complete datasets, such as image recognition [MPR+12]. We can also find some generic and commercial approaches with which datasets can be generated using a web application, program, or downloadable framework that provides an API.

An overview of some popular tools can be seen in Tables 7.2 and 7.3. A couple of interesting examples are *GenerateData.com*, which is a web site (and open source distribution) to generate generic (domain-independent) datasets, and *Mockaroo*, which is a service that provides a large number of editable parameters to build a broad variety

of data collections for different purposes. Most of these tools focus on the generation of data that can be used to fine-tune databases or to obtain sample data, rather than on the generation of data to evaluate recommender systems or context-aware recommender systems. Moreover, none of them is able to represent and generate information about context. We could potentially use one of those tools to generate for example datasets of synthetic items; for instance, with Mockaroo we could define attributes for the Iris dataset (see Section 8.3.2.1), except in the case of the *class* attribute, as Mockaroo does not allow specifying the need to select a value from a fixed set of words, like in this case the set of available class names.

Tool	References	Purpose	Type	Developed in
Databene Benerator	[Ber91, ARMCM13]	Perform realistic load and performance tests	Framework	Java
GeCo	[TVC12, TVC13]	Evaluate algorithms in contexts where realistic personal data are required but privacy concerns prevent the use of real data	Framework, web application	Python
Generate-data.com	[Kee10]	Test software, populate databases, etc.	Web application	JavaScript, PHP and MySQL
Mockaroo	[Bro16]	Test software and develop demo software	Web application	Ruby
DBMonster	[Maj12]	Test SQL databases	Framework	Java
Fake Name Generator	[Cor16]	Test software and other purposes where fake personal data may be needed	Web application, program	C++ (SourceForge version)
IBM Quest Synthetic Data Generator	[IBM15, RA+94, CK14]	Mine associations and sequential patterns in transaction data	Framework	C++
DataGen-CARS	[dCRHIHTL17a]	Generate synthetic datasets for the evaluation of context-aware recommendation systems	Framework	Java

Table 7.2: Overview of some existing dataset generators: basic information.

In any case, the tools shown in Tables 7.2 and 7.3 would not be suitable to generate datasets for the evaluation of CARS, and we would miss some interesting features offered by DataGenCARS (described in Chapter 5), like the possibility to define profiles of items (thus supporting the definition of certain correlations among attributes), define profiles of users (with the definition of their preferences or utility functions), complete existing datasets (e.g., replacing null values or adding context attributes), extract statistics from existing real data, generate ratings considering user preferences, mix real and simulated data, extract real data from external data sources like OpenStreetMap, simulate noise/uncertainty, and map automatically from schemas to Java classes and vice versa, to cite some examples.

Tool	Attribute Generators	File Formats	Maximum number of tuples	Fits to CARS
Databene Benerator	Random, cumulated, increment, random walk, shuffle, wedge, bit reverse, expand, head, literal, numbers with weights, Fibonacci, Padovan	CSV, DbUnit, Excel, Flat, SQL, Text, XML, Custom	Unlimited	No
GeCo	Random, normal, uniform	CSV	9999, unlimited with the non-Web version	No
Generate-data.com	Random, normal	CSV, Excel, HTML, JSON, LDIF, JavaScript, Perl, PHP, Ruby, SQL, XML	100 (free), 5000 ($20/year), unlimited with the non-Web version	No
Mockaroo	Random, Poisson, normal	CSV, Tab-Delimited, SQL, Excel, JSON, DBUnit XML	1000 (free), 10000 ($50 per year), 10M ($500 per year)	No
DBMonster	Random	SQL	Unlimited	No
Fake Name Generator	Random	CSV, Excel, HTML, SQL, text	50000, unlimited with the non-Web version	No
IBM Quest Synthetic Data Generator	Random, Poisson, exponential	SQL	Unlimited	No
DataGen-CARS	Random, uniform, and Gaussian	CSV, text	Unlimited	Yes

Table 7.3: Overview of some existing dataset generators: features.

In the field of recommendation systems in particular, we can find some interesting approaches proposed to bridge the gap between algorithm design and evaluation. Most of the authors working on RS evaluate their recommendation algorithms by using well-known datasets, such as MovieLens [Gro16], Flixster [ZL09], or Book-

Crossing [Zie04], among others. The use of well-known collections have brought about an unexpected side effect, as many researchers design their proposals focusing the evaluation processes only on achieving a reasonable good performance on those specific datasets (in order to compare their approaches with the ones developed by other authors), but not testing the performance under different conditions.

In order to overcome this problem, some attempts to design synthetic datasets have been made. In [TST06b, TST06a], the authors put forward a methodology to generate synthetic datasets for evaluating RS. However, they only focus on the generation of item attributes. Moreover, their work does not take into account the generation of data regarding the context, and therefore the collections generated by using that approach could not be used to evaluate CARS. Some other authors, as is the case of Wang et al. [WTH10] and Christakopoulou and Karypis [CK14], designed ad-hoc data collections to simulate the behavior of customers in a real environment to better test their algorithms. Both approaches use the *IBM synthetic dataset generator* [AS94], originally built to simulate user transactions for testing databases. This tool supports the parametrization of the size of the set, as well as the average number of items per transaction, or even the introduction of some repetitive patterns into the sample. However, it does not offer any functionality to generate the context in which each recommendation or rating takes place.

In the field of CARS, there exist some interesting works on synthetic data generation. So, in the work by Pasinato et al. [PEAZ13] an abstract methodology to build context-aware collections of data (in terms of item ratings and context attributes) was presented. However, they only sketch the steps to build this type of generators, leaving a number of issues open, such as how to introduce noise in the ratings generated. Furthermore, they do not carry out an empirical evaluation to validate or illustrate the possibilities of their approach. Another work in this line is the one by Lee and Kwon [LK14], who presented a TV content RS, where they generated random contexts for evaluation purposes. This is an example of ad-hoc data generation that cannot be used in other domains, such as domains in which the GPS location of the user is essential.

7.6 Summary of the Chapter

In this chapter, we presented relevant works in the area of context-aware recommendation systems. First, we discussed several approaches that introduce the context dimension during the recommendation process. In order to facilitate the understanding, we organized the studied works in the categories of pre-filtering, post-filtering, and contextual modeling. We also described some frameworks and examples of context-aware recommendation systems for different domains. Second, we explained some context-aware recommendation approaches applied to mobile computing. Specifically, we focused on the pull-based and push-based algorithms. Several examples of context-aware mobile recommendation systems were also exposed. Third, we described related work on location-aware recommendation systems for mobile environments. In addition, we presented the main applications of this type of systems in different rec-

ommendation scenarios. Then, we discussed some recommendation approaches for mobile users based on P2P communications. Finally, we analyzed a set of tools that facilitate the generation of synthetic data: the datasets resulting from these tools are not appropriate for the evaluation of context-aware recommendation systems, which motivated the development of our DataGenCARS tool.

Chapter 8

Experimental Evaluation

In this chapter, we present the experimental evaluation performed. In Section 8.1, we first present the prototype of the MOONRISE framework. Specifically, we show the main modules of the architecture developed, to address the existing gap between CARS and mobile computing. In Section 8.2, we present some experiments focused on analyzing the potential interest of context-aware recommendations. In Section 8.3, we perform a set of experiments to evaluate DataGenCARS. In Section 8.4, we show the experimental results of the evaluation conducted in the use-case scenario described in Chapter 6, where a context-aware mobile recommendation system suggests works of art in real time to a user in a museum. Besides, in Appendix C, we evaluate two alternative solutions (one based on the use of the Hidden Markov Model and another one exploiting Information Retrieval techniques) to the problem of identification of the type of item that the user specifies in a pull-based recommendation system. All the experiments were conducted on a computer with an Intel Core $i5-2320$ processor with 3 GHz and 16 GB of RAM, running Windows 7.

8.1 Prototype of the Recommendation Framework

We have developed a prototype of our framework, whose aim is to facilitate the development of context-aware recommendation systems in mobile environments. This prototype is a Java library (Java 1.8 was used) that uses the following application programming interfaces (APIs):

- *Apache Mahout 0.9* [Apa14]. It allowed us to include collaborative filtering recommendation algorithms (e.g., user/item-based recommendations, SVD, etc.), random-based recommendations, as well as extend and implement 2D recommendation algorithms (e.g., content-based recommendations and popularity-based recommendations) and context-aware recommendation algorithms (e.g., pull-based recommendations that use the pre-filtering, post-filtering and contextual modeling paradigms), respecting the recommendation hierarchical structure of Apache Mahout.

- *Weka data mining toolkit 3.7.6* [Uni93]. It was used for the implementation of the contextual modeling recommendation paradigm, that includes a classification algorithm.
- *SQLite JDBC driver 3.14.2* [Sai16]. It was used to manage SQLite databases from Java code.
- *Apache Lucene 2.4.0* [Apa05]. It was used for the pre-processing of keywords introduced in the system, as well as for the indexing of the documents for an IR-based method, that identifies the type of item required by the user in a pull-based recommendation (see Section 4.1.6.2), by using keywords.
- *Jahmm 0.6.1.* It allowed us to use the Hidden Markov Model functionalities for the implementation of the HMM-based method that identifies the type of item from keywords (see Section 4.1.6.1) as in the IR-based method.

In Chapter 3, we described the generic design of our propose architecture. In our prototype, we have implemented the following modules of the logical layer: *Repository Manager*, *User Profile and Context Manager*, and *Pull-Based Recommendation*. In order to facilitate the maintenance and understanding of the prototype structure, the code was finally organized into nine modules (or packages). In Figure 8.1, we show the package diagram of the prototype according to our current implementation (e.g., the push-based recommendation package is still missing in our prototype).

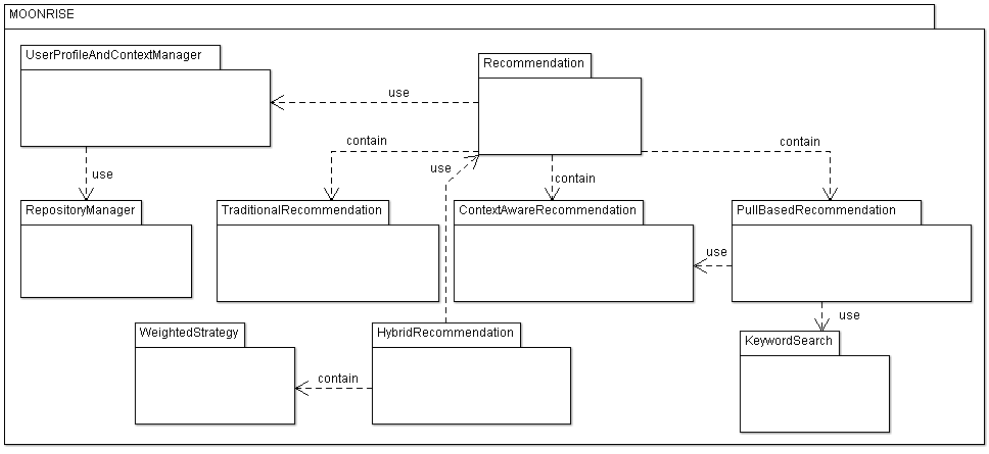


Figure 8.1: Package diagram of the MOONRISE prototype.

As an example, we show the class diagram of the *ContextAwareRecommendation* module in Figure 8.2. Other class diagrams and a description of the main classes are included in Appendix A.

Moreover, from the proposed context model (see Section 3.2.3), we generate the Entity-Relationship (E/R) schema shown in Figure 8.3. This schema describes in

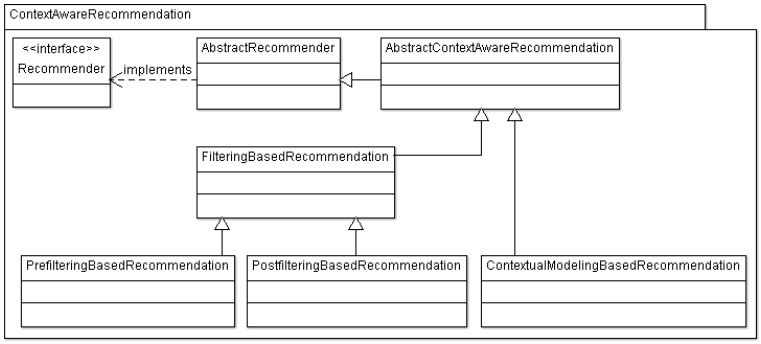


Figure 8.2: Class diagram of the Context-Aware Recommendation module.

an abstract way the representation of the persistent data used by a context-aware recommendation system.

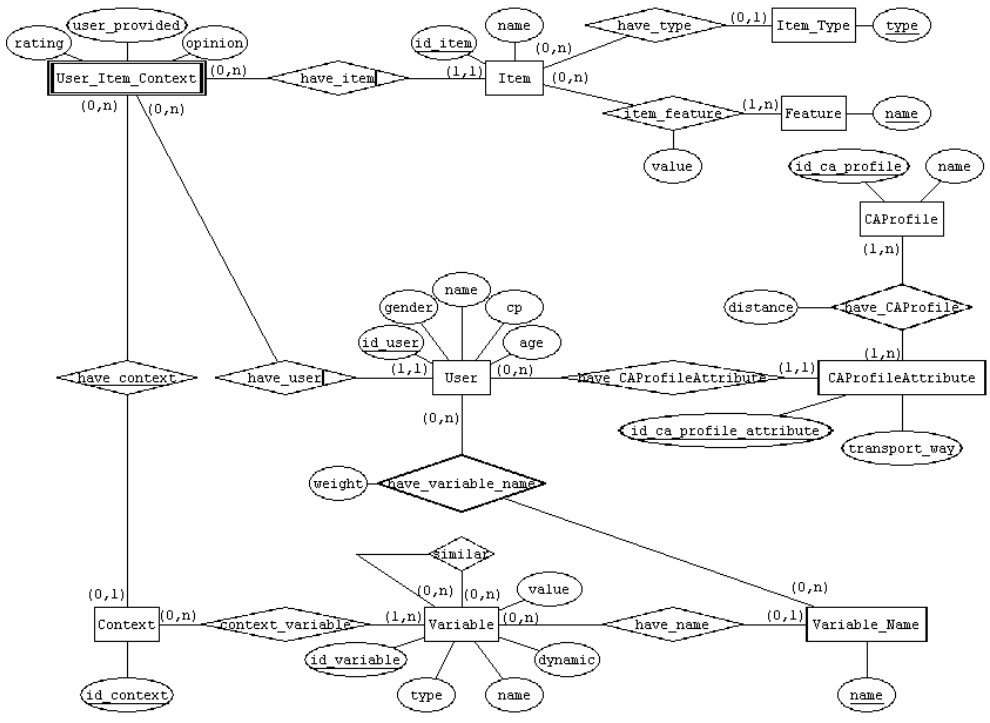


Figure 8.3: Entity-Relationship model for a context-aware recommendation system.

8.2 Experimental Evaluation of Mobile Contextual Recommendations

In this section, we present an experimental evaluation focused on mobile context-aware pull-based recommendations. In Section 8.2.1, we describe the dataset chosen for the evaluation. Then, we analyze the identified limitations in the dataset. In addition, we explain the adaptation performed on the dataset for our experimental evaluation. In Section 8.2.2, we present the experimental settings. In Section 8.2.3, we show some experimental results applying soft and hard constraints. Besides, we evaluate the impact of the sparseness of the dataset. Finally, we present a global discussion of the results obtained.

8.2.1 Analysis of the Dataset Used (STS Application)

A very important problem to evaluate mobile CARS is the lack of available datasets that contain information about the context of the user when she/he provided a specific rating. It would be interesting to have some reliable high-dimensional dataset (e.g., like the well-known Movielens [Gro16]), but enriched with precise and complete contextual information. Unfortunately, the current situation is quite different:

- Most papers in the field of context-aware recommendation systems for mobile environments have been evaluated with own data collected from real users (e.g., see [WHBGV11, LCVA12, LWGD07, RDB+08, BBC+08, YZZ+06, WRW12, WSW07, BNWP11]). For example, in [WRW12], to evaluate a context-aware mobile music recommendation system, the authors firstly considered the publicly available CAL500 dataset [TBTL07], which incorporates ratings along with some usage annotations (such as whether the user was driving or sleeping). However, those annotations only partially covered the evaluation needs of the authors, and so they decided to build a new dataset with songs crawled from Grooveshark [BGT06] and YouTube [HCK05] and using humans to annotate them.

- There are some context-aware datasets available in the Internet that contain contextual information (e.g., see [BGT15]), in domains such as foods and restaurants [ONMU06, OTMA09, RGGV14], movies [ASST05], music [BKL+11], and hotels and travel [ZBM12, ZMB14, EBRT13, BERS13]. There is also a Yelp's Academic Dataset [Yel14, Asg16], which includes ratings provided by real users to score local businesses (e.g., in [EVMT12] these data were enhanced by simulating the context). ConcertTweets dataset [Ada14, AT14] is another interesting dataset, which contains concert ratings combined with spatio-temporal contextual dimensions and data of social networks.

So, an important current problem in this area is the lack of a good-quality large dataset with enough information on ratings and contexts for (quantitative and qualitative) evaluation purposes of context-aware mobile recommendation paradigms and applications developed by the research community.

Of the few datasets available, the one most suited to our problem, and therefore the one that we decided to use, was the data collected by an Android Mobile Application called *South Tyrol Suggests* (*STS*) [BER13b, BER14]. The STS application provides context-aware suggestions for accommodations, attractions, events, and restaurants in South Tyrol. Specifically, the dataset available at [BER13a] contains 2534 ratings (on a scale of one to five), provided by 325 users in different contexts, of approximately 249 point of interest [EBRT13, BERS13]. Contexts in the dataset are composed by 14 context dimensions:

- *User context* (8 variables): transport way (walking, bicycle, car, public), week of the day (weekday, weekend), mood (happy, sad, active, lazy), companion (alone, friends, family, girlfriend, children), time available (half day, one day, more than one day), knowledge of the surroundings (new to the area, returning visitor, citizen), travel goal (visiting friends, business, religion, health care, social event, education, landscape, fun, sport), and distance to the item (far away, nearby).

- *POI context* (2 variables): budget (budget traveler, price for quality, high spender, free) and crowdedness (crowded, not crowded, empty).

- *Environment context* (4 variables): season (spring, summer, autumn, winter), temperature (burning, hot, warm, cool, cold, freezing), time of the day (morning, noon, afternoon, evening, night), and weather (clear sky, sunny, cloudy, rainy, thunderstorm, snowing).

The values of context variables related to the environment were automatically collected, whereas the values about the context of the items (POIs) and the user context were manually obtained (the application requested the user to explicitly fill appropriate values for those variables when submitting her/his rating), with the exception of the context variable weekday, which was automatically computed. From now on, we will consider 13 of the previous 14 context variables, as the distance between the user and the item (considered as a variable of the user context) will be translated into a geographic location for the item.

Analysis of the Dataset

Although the dataset chosen is the most appropriate one that we have been able to find for the evaluation of recommendation systems that suggest POIs, it is subject to important limitations. A notable problem is that it is pretty sparse in terms of the availability of ratings and contexts. Figure 8.4 illustrates the low percentage of information available for each context variable in the full dataset, which emphasizes the interest of techniques that capture contextual information automatically without the intervention of the user. As shown in the figure, the context variable for which there is more information available is the "temperature" variable (a value of temperature is provided in about 15.6% of the ratings), whereas the variable with more missing values is the "transport way" (specified only for 3.2% of the ratings). The percentages

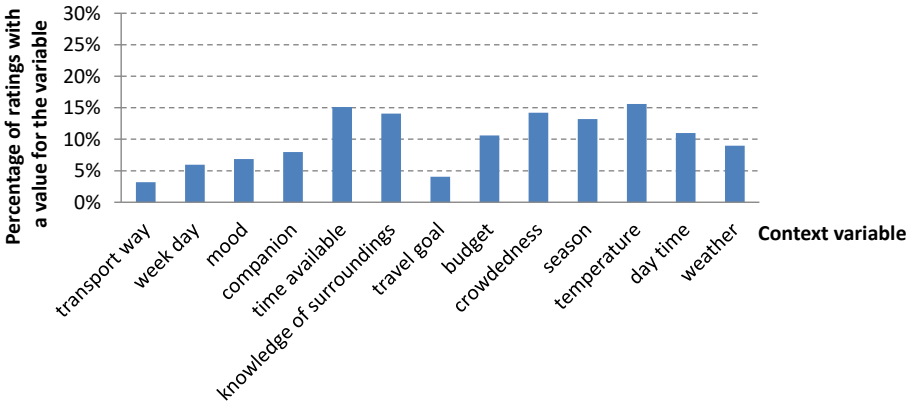


Figure 8.4: Analysis of the STS dataset: percentage of ratings with a value for each context variable.

shown in the figure are really low, which means that in most cases the real value of a specific context variable potentially affecting a rating is not available.

This sparseness of contextual information is further analyzed in Figure 8.5, which indicates, for each rating, the number of context variables that have a value. As the figure shows, only a maximum of 5 context variables are provided for some ratings and most ratings have only 1 context variable defined (or none). For 23.64% of the ratings in the dataset only one context variable has a defined value and 38.44% of the ratings have no context variable defined.

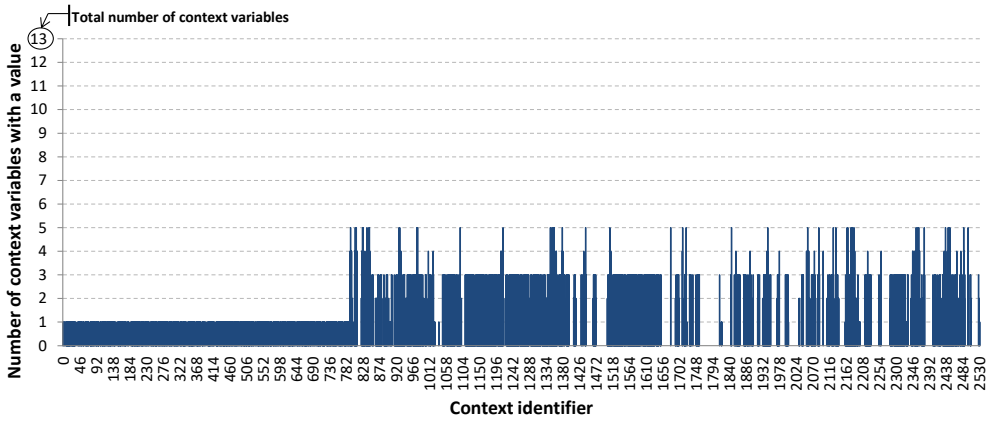


Figure 8.5: Analysis of the STS dataset: amount of variables with a value for each context.

The ratings available are not uniformly distributed over the different users either: for some users there are very few ratings available and for others we have compara-

tively a high number of ratings. So, there is an average of 8 ratings per user, but some users have provided many ratings (for example, a single user provided 175 ratings) and some users provided only 1 rating.

As a reasonable decision could be to consider only the users for which a significant number of ratings is available, we analyzed the contextual information for the three users with more ratings in the dataset: user 1, user 7, and user 24, who have 175, 96, and 123 ratings, respectively; to determine an appropriate number of users to select, we applied the clustering algorithm k-means for different values of k. Figure 8.6 shows, for those three users, the number of ratings with no contextual information, the number of ratings with some contextual information, and the number of contexts guaranteed to be different (two contexts that have the values of all the defined context variables equal but that also have some undefined values are not guaranteed to be different or equal). Just by coincidence, these three users voted exactly the same items (numbered from 1 to 8 in the figure). Each division/portion of each bar in the figure marks a value in the Y-axis. For example, the figure shows that for user 1 and item 1: the item was rated by the user 43 times; of those, 13 ratings completely lacked contextual information and 30 ratings had some contextual information (i.e., at least 1 context variable had a defined value); moreover, it also shows that, from the ratings with some contextual information, only 23 contexts are guaranteed to be different. It should be noted that in some cases the portion corresponding to the number of ratings with some contextual information is hardly appreciable (e.g., for item 5 of user 1, for item 6 of user 7, etc.) because that number matches the number of contexts guaranteed to be different. The figure shows that the sparseness of contextual information also holds in the case of the users for whom a higher amount of information is available. Notice that the number of ratings of each user is in general higher than the number of items that she/he rated. This is because there are users that have rated the same POI several times even in potentially-identical contexts (i.e., identical contexts if we ignore context variables with undefined values).

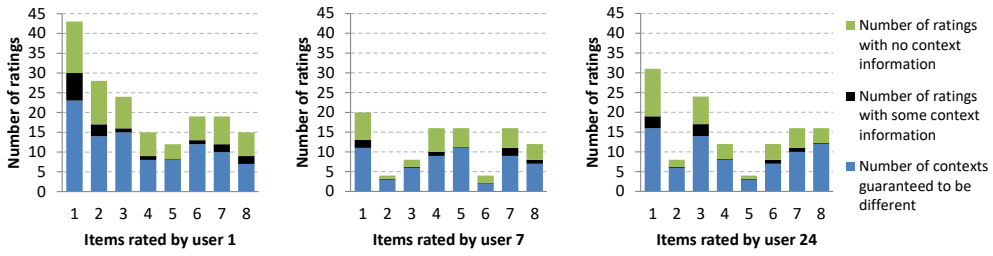


Figure 8.6: Analysis of the STS dataset: contextual information of the items rated by some selected users.

Adaptation of the Dataset

As the location is an important context parameter, that is not available in the dataset, we decided to adapt the original dataset by including a new context variable to represent the location (location of the user in the case of the user context, location of an item in the case of the item context). For this purpose, we completed randomly the values of the context variables "distance to the item" and "transport way". Firstly, we generated randomly the location values (latitude and longitude) of the 249 POIs, assuming that they are bounded by an area of 3 Km^2 (i.e., the density of POIs is about 83 per Km^2). Then, we simulated the locations of the users in a certain context by taking into account several user profiles (see Table 8.1) –each user is assumed to belong to one of those profiles–, the values of the context variable "transport way" (as there are missing values for this context variable in the dataset, we generated the missing values randomly), and the locations of the POIs generated previously. Although these adaptations were performed without using DataGenCARS (see Chapter 5), it should be noted that with DataGenCARS we could also complete the missing values of the context variable "transport way" and generate the "locations" of the POIs.

As an example, according to Table 8.1, if a user with a "lazy profile" is driving in her/his car, then she/he will consider that those POIs at a distance exceeding 1 Km are located "far". Now, let us imagine that we have a rating provided by this user for a certain item and that the user indicated that the item was "near" (the location she/he was at when she/he decided to visit that POI). In order to simulate a plausible location of the user, we proceed as follows:

1. We generate randomly a location within a scenario of 4 Km^2. This size is slightly higher than the area of 3 Km^2 where the items are positioned, in order to simulate the potential absence of items, but not of users, near the borders of the scenario.

2. We compute the distance between the location of the user and the POI. If the distance is smaller than 1 Km (the maximum radius for that user profile and transportation means), then that location is valid for the user's rating and can be kept, and otherwise we get back to step 1.

		Transportation means			
		Walking	Bicycle	Car	Public
Profile	lazy	0.50 Km	0.70 Km	1.00 Km	0.85 Km
	normal	1.25 Km	1.35 Km	1.50 Km	1.42 Km
	active	2.50 Km	2.70 Km	3.00 Km	2.85 Km

Table 8.1: User profiles and their maximum distances considered for nearby items.

As we are generating some data randomly to complete the dataset, it is important to ensure that this artificial addition does not modify any substantial aspect of the

dataset. For that purpose, we computed the Pearson correlation coefficient to determine the relationship between the ratings provided by the users and the distance to the corresponding POIs. We obtained a Pearson coefficient of 0.18 (very slight positive correlation) for the original values in the dataset, which indicates that the user does not generally penalize ratings for items that are far. For the simulated data, we obtained a coefficient of -0.01 (near independency, which is consistent with the fact that the missing data were generated randomly). In both cases, there is a very weak correlation. Therefore, the modifications to the dataset do not affect the real correspondence between the ratings and the distances between the users and the items rated. In Figure 8.7, we show the distribution of ratings for each real distance indication (item near the user or far from the user) and we can see that there is a high percentage of high ratings (rating values of 4 and 5) for both of those two distance indications, not only for nearby items.

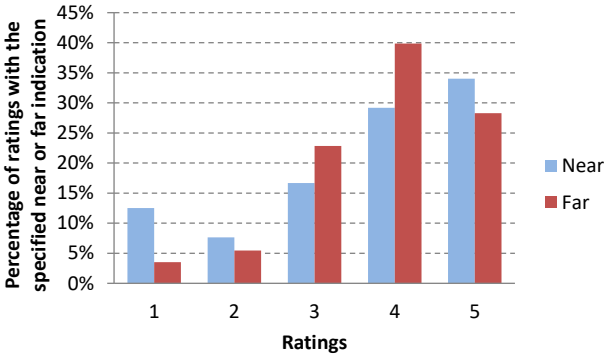


Figure 8.7: Percentage of ratings with a certain distance indication (near/far).

8.2.2 Experimental Settings for the Evaluation of Context-Aware Recommendations

In order to evaluate the performance of the context-aware recommendation paradigms included in the prototype, we compared the pre-filtering, post-filtering, contextual modeling, and traditional recommendations. For that purpose, we have used the classical evaluation metrics of *MAE*, *precision*, *recall*, and *F-measure* (see Section 2.4.1). The experiments were performed with the dataset STS, described, analyzed and adapted in Section 8.2.1. Our experiments include a training phase and a testing phase. Hence, we decided to divide 70% of the data for *training* and 30% for *testing*. The training set represents information available about the user profiles (previous ratings already provided by the users), whereas with the testing set we simulate items that have not been rated yet (the system will try to predict the right recommendations for those items). The experimental settings used for evaluation are shown in Table 8.2.

Parameter	**Default Value**
Amount of data for training	70%
Amount of data for testing	30%
Traditional recommendation algorithm	SVD
Classification algorithm for contextual modeling	Naïve Bayes
Similarity threshold	0.5
Recommendation threshold	3

Table 8.2: Experimental settings for the evaluation of context-aware recommendations.

Firstly, we compare the *traditional* non-contextual collaborating filtering with the context-aware paradigms of *pre-filtering*, *post-filtering*, and *contextual modeling*. Due to the limited information available in the dataset, we decided to use the SVD recommendation algorithm offered by Apache Mahout [Apa14] for the *pre-filtering* and *post-filtering* paradigms. The SVD recommendation model allows a better handling of the problem of *cold start*, by capturing indirect relationships between users and items (i.e., it is able to relate several users even when they have not rated items in common). For the *contextual modeling*, we use the Naïve Bayes classification algorithm provided by Weka [Uni93].

In the *pre-filtering* and *post-filtering* paradigms, the similarity threshold was set at 0.5 and the similarity between two contexts was calculated by using Equation 4.1 (see Section 4.1.2). In the *pre-filtering* paradigm, for the computation of context similarities by Algorithm 2 (see Section 4.1.3), we apply for *compareVariablesWithMissingInfo* the strategy that ignores context variables with unknown information and assumes a maximum similarity of 1 when a variable has a value and the other does not. We made this optimistic decision because the information of context vectors is very poor (there is a high sparseness in the representation of contexts, as explained in Section 8.2.1), and otherwise we would filter out too many contexts.

In the *post-filtering* paradigm, we implemented the *filteringWithSoftConstraints* algorithm called in Algorithm 4 (see Section 4.1.4) by penalizing the predicted rating for the case of items that are far from the user, by measuring by how much the distance of what is considered near is exceeded (according to the user profile and her/his transportation means, as shown in Table 8.1). For this purpose, we verified if the rating predicted is greater than or equal to the recommendation threshold (set to 3). If so, we penalize the predicted rating by using Algorithm 11, where NNR is the number of not recommended rating levels (integer rating values below the minimum required threshold). It should be noted that the fact that from the results shown in Figure 8.7 can be inferred that users do not seem to penalize distant items does not imply that reaching such items does not have a cost to the user (the user only values the items that she/he decides to visit).

Algorithm 11: POST-FILTERING: PENALIZATION OF THE RATING PREDICTED BASED ON THE DISTANCE

Input: The radius indicating what is considered near for the corresponding user profile and transportation means ($radius$), the distance between the user and the item ($distance$), and the minimum recommendation threshold required ($minRecommendThreshold$).

Output: The rating predicted (RP) after the penalization is applied.

```
1  RP ← 1;
2  NNR ← minRecommendThreshold − 1;
3  excessDistance ← distance − radius;
4  penalizationIncrement ← NNR/3;
5  if (excessDistance < radius/NNR) then
6  |  RP ← minRecommendThreshold − (1 ∗ penalizationIncrement);
7  else
8  |  if (((radius/NNR) ≤ excessDistance) ∧ (excessDistance <
   |    (2 ∗ (radius/NNR)))) then
9  |  |  RP ← minRecommendThreshold − (2 ∗ penalizationIncrement);
10 |  else
11 |  |  if (((2 ∗ (radius/NNR)) ≤ excessDistance) ∧ (excessDistance ≤
   |  |    (3 ∗ (radius/NNR)))) then
12 |  |  RP ← minRecommendThreshold − (3 ∗ penalizationIncrement);
13 return RP;
```

8.2.3 Experiments Comparing Traditional and Context-Aware Recommendation Paradigms

We distinguish the experimental results according to whether only *soft constraints* are applied (relevant context variables are marked, by assigning them an appropriate weight) or whether *hard constraints* (strict conditions on context variables that need to be satisfied) are also required. The details of these constraints are provided in Sections 4.1.3 and 4.1.4.

Experimental Results with Soft Constraints

Figure 8.8 compares the different paradigms in the case of soft constraints. Regarding the MAE, we can see that the traditional method achieves very similar results to the pre-filtering paradigm, being these two the paradigms that provide the smaller MAE. Figure 8.8 shows that the precision of the pre-filtering and the traditional paradigms perform similarly to the post-filtering, and that contextual modeling performs slightly better. In terms of recall and F1-measure, the pre-filtering and traditional paradigms also perform similarly and outperform the post-filtering and contextual modeling paradigms. Specifically, we can observe that with the post-filtering and contextual

modeling paradigms the recall decreases in about 45.8% and 18.2%, respectively. Similarly, the F1-measure decreases in about 26.7% with post-filtering and about 7.9% with contextual modeling. We think that an important reason for not observing very large differences with the traditional paradigm is the lack of dense contextual information in the dataset used.

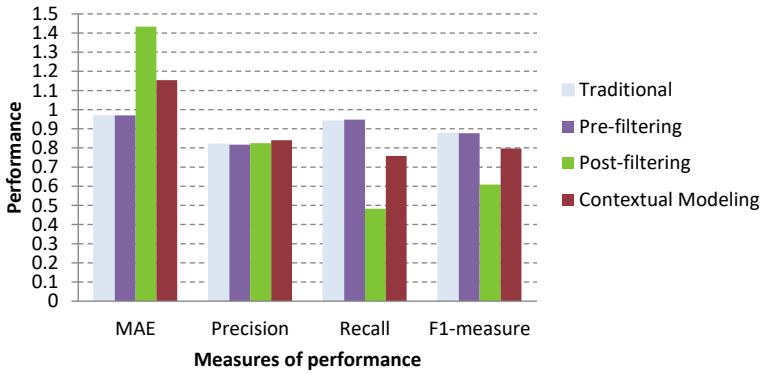


Figure 8.8: Comparison of paradigms with soft constraints: MAE, precision, recall, and F1-measure.

As indicated in Section 8.2.1, the users with more ratings in the dataset are the users 1, 7, and 24, with 175, 96, and 123 ratings, respectively. For these users, we also determined the specific values of the *MAE* and *F1-measure* that apply to them (see Figure 8.9 and Figure 8.10). We can see that there is a significant variability regarding the performance of the different methods for different users, as the quality of the recommendation models is heavily influenced by the amount of data available for those users. In general, the more data available (i.e., the higher the number of ratings per user) the better the accuracy of the recommendations.

To facilitate the analysis of the existing relation between the number of ratings available and the recommendation performance, in Figure 8.11 we show the MAE achieved for each of the recommendation methods when the number of ratings varies. This figure presents results based on experimental data for the users 1, 7, and 24, but a similar trend can be observed if we consider other users. For users with a very low number of ratings the training phase has a very limited amount of data available to build the recommendation model and, consequently, the recommendation performance decreases. Later in this section, we analyze the impact of the problem of sparseness of the dataset in detail.

Experimental Results with Hard Constraints

Now we consider the case of hard constraints. As an example, we consider as a hard constraint the existence of a location-dependent constraint that requires that an item must be near the user in order to be recommended. The concept of nearness is

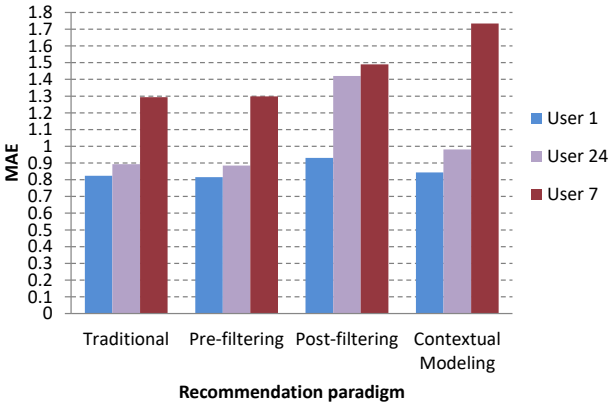


Figure 8.9: MAE for users 1, 7, and 24 (soft constraints).

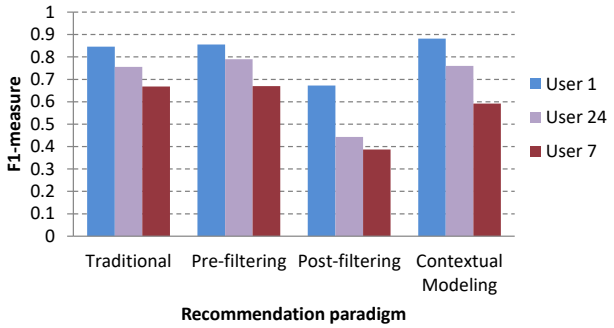


Figure 8.10: F1-measure for users 1, 7, and 24 (soft constraints).

relative: it depends on the user profile and the transportation means used, as shown in Table 8.1.

We compare a traditional recommendation algorithm with the post-filtering paradigm. In the case of applying hard constraints, the traditional paradigm is clearly worse in terms of precision and F1-measure (see Figure 8.12). The reason is that the post-filtering paradigm is able to remove items that do not satisfy the hard constraints, which are not removed by a traditional recommendation algorithm, as it does not apply any kind of post-filtering. Removing irrelevant items increases the precision (by a factor of two, according to Figure 8.12). As the post-filtering does not remove relevant items (only the items that do not satisfy the hard constraints are removed), it does not affect the recall. As a result, the overall F1-measure improves considerably. So, we see that checking hard constraints in a post-filtering step is essential.

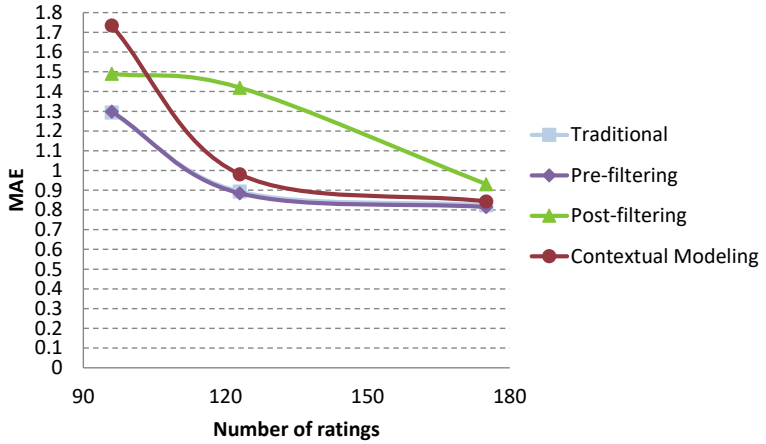


Figure 8.11: Relation between the MAE and the number of ratings available (users 1, 7, and 24).

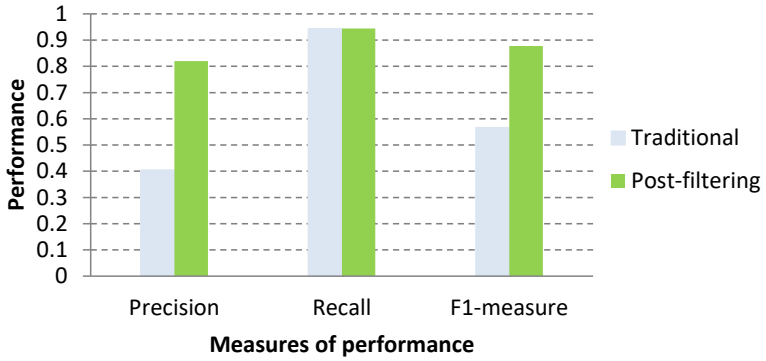


Figure 8.12: Comparison of the traditional and post-filtering paradigms with hard constraints.

Impact of the Sparseness of the Dataset

The little information that we have in the dataset is considered as a reason for performance reduction in the learning of an appropriate recommendation model. As a result, the prediction of some ratings might be unsuitable, which in turn could lead to imprecise or incomplete recommendations.

We have carried out an experiment to prove the hypothesis that the more data available the better the learning model will perform. Firstly, we discarded the users for which a very low number of ratings were available; specifically, we required at least 11 ratings per user, which led us to the selection of 24 users (from a total of 325 users). Then, we divided the dataset randomly in two subsets: 70% of the

ratings of each user were used for training and the remaining 30% were used for testing. Afterwards, we removed from the testing set those items and users that never appear in the training set (neither jointly nor as part of different ratings). From the above training set, we temporarily generated another "temporal training set", where we initially added only one rating per user. The rest of the ratings were stored in a "pool set". Next, we trained the recommendation model with the temporal training set and we evaluated it with the testing set focusing only on the users 1, 7, and 24. Later, we incrementally added to the temporal training set a maximum of 10 ratings per user by using certain addition strategies:

- *Option 1:* adding ratings from the pool set to the temporal training set one by one until a maximum of 10 ratings are added.
- *Option 2:* adding 5 ratings from the pool set to the temporal training set and then 5 ratings more.
- *Option 3:* directly adding 10 ratings from the pool set to the temporal training set.

We learned an SVD recommendation model using Apache Mahout with the complete training set and performed the evaluation with the testing set; this evaluation was named *baseline*, and it represents the model that can be obtained using all the information available in the existing dataset, which means a total of 649 ratings in the training set. After that, we trained an alternative SVD model with the temporal training sets that were generated with the option 1, option 2, and option 3, and we evaluated the MAE, precision, and recall of each variant with the testing set. Notice that each option implies a different number of training-testing stages (11, 3, and 2, respectively), with an increasing number of ratings in the temporal training set used in each stage. Figures 8.13, 8.14 and 8.15 display the results obtained. The figures clearly show that the quality of the model increases when more training data is available. Therefore, we could expect that better recommendations could be obtained if a richer dataset was available.

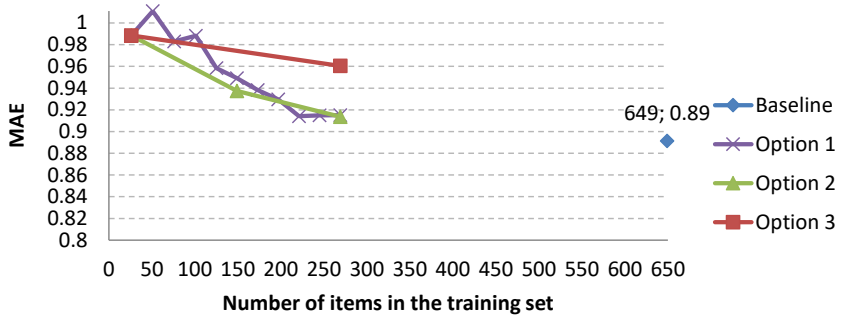


Figure 8.13: Evolution of the MAE with more training data.

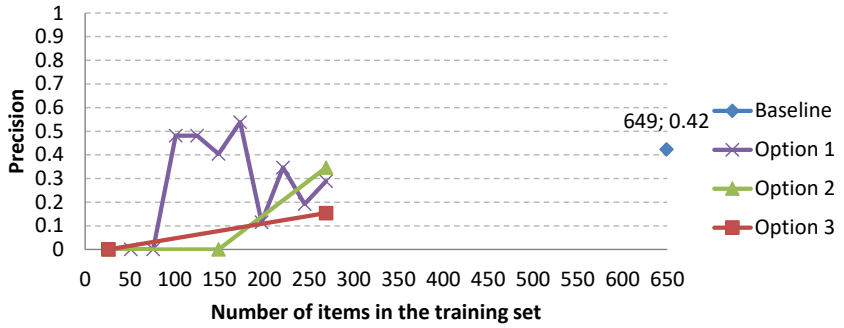


Figure 8.14: Evolution of the precision with more training data.

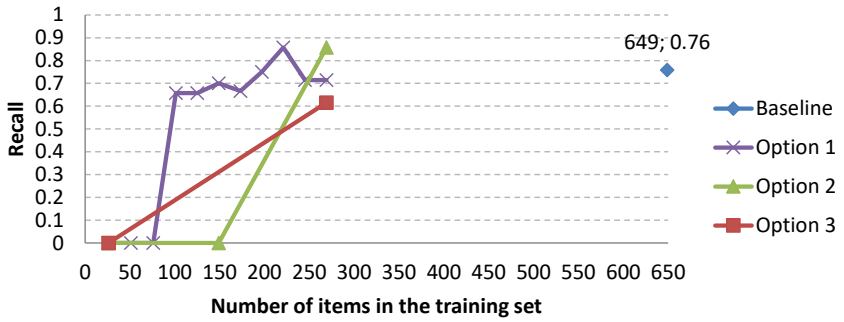


Figure 8.15: Evolution of the recall with more training data.

We can illustrate with some data the effect observed in the previous figures. For example, if we focus on the MAE, we can see in Figure 8.13 that when there is initially one rating for each user the value of the MAE is 0.99. From this initial training set, for *Option 1*, when two ratings per user are added (i.e., there is a total of 76 ratings in the training set) the value of the MAE decreases slightly to 0.98, and so on until a value of 0.91 is obtained when there are 269 ratings in the training set. For *Option 2*, when five ratings per user are added (which leads to a total of 149 ratings in the training set) the value of the MAE decreases from 0.99 (in the case of one rating per user) to 0.94, and when adding five ratings more per user the MAE decreases to 0.91. Finally, for the *Option 3*, when ten ratings are added per user (i.e., there is a total of 269 ratings in the training set) the MAE is reduced to 0.96. In the previous three options, by increasing the number of ratings per user in the training set, the values of the MAE tend to decrease to values around 0.89 (the value obtained for the baseline when the training set contains 649 ratings in total, that is, using the whole dataset). A similar behavior can be observed for the precision and recall measures (Figures 8.14 and 8.15). The key aspect that should be noted is that there is a clear trend showing that the performance can increase when more training data (i.e., historical data about previous users' rating) increases.

Discussion About the Results Obtained

The experimental evaluation showed a couple of results that were unexpected:

- When hard constraints are used, a post-filtering step can significantly improve the performance.
- With the STS dataset, when only soft constraints, we cannot observe a clear improvement of the other paradigms over the traditional recommendation paradigm. We believe that the reason for this is the low quality of the dataset: even if it is the best one that we have found, it has a significant amount of missing information and poor context descriptions (see Section 8.2.1). As a result, the models learnt are not precise enough. A better context-rich dataset is expected to significantly improve the results and highlight the differences, as other experiments that we have performed show (e.g., Sections 8.3.1.1 and 8.4.2).
- The recall with the post-filtering paradigm on the STS dataset decreases significantly (see Figure 8.8). This is because the post-filtering penalizes data items that are far from the user; indeed, the difference between the post-filtering paradigm and the traditional paradigm is the application of this penalization *a posteriori* (as shown in Algorithm 4). Whereas this seems reasonable, according to our analysis in Section 8.2.1, with the dataset STS considered, the distance between the user and the item does not seem to have a direct key impact on the final rating provided by the user. Nevertheless, in Section 8.2.3 we actually showed how the post-filtering can significantly increase the accuracy of the recommendations when there are hard constraints regarding the maximum distance allowed (see Figure 8.12).

In summary, we conclude that the context may play a key role but that, at the same time, a major inconvenience for the experimental evaluation of context-aware recommendation paradigms is the difficulty to find a suitable dataset that incorporates contextual information.

8.3 Experimental Evaluation of DataGenCARS

In this section, we explain the experimental evaluation that we have performed to show the usefulness of DataGenCARS, described in Chapter 5. We present two sets of experiments. On the one hand, the goal of the experiments presented in Section 8.3.1 is to show the usefulness of generating a completely-synthetic dataset to evaluate context-aware recommendation algorithms. On the other hand, the experiments shown in Section 8.3.2 focus on the capabilities of DataGenCARS that support the generation of realistic datasets. In both cases, we carry out an evaluation with an offline setting [SG11, HKTR04].

In some of the experiments, we evaluate a context-aware recommendation algorithm and a traditional recommendation algorithm that does not take the context

factors into account. In particular, we compare an approach of *Contextual Modeling* [ASST05, AT08], using a classification algorithm based on Naïve Bayes [JL95], with a traditional user-user collaborative filtering algorithm based on *SVD* [BP98, SK09]; in both cases, the class of items to recommend contains the items whose predicted rating (from one to five) is above a threshold of three. Even though Naïve Bayes is based on the assumption that variables are independent, which is rarely the case, several studies have shown that it behaves surprisingly well in practice, as the classification decision may be correct even if the probability estimates are not accurate [LIT92, DP97, Ris01, Zha04, HTF01, HY01, Zha05]. Nevertheless, these algorithms have been selected for illustrative purposes. It should be noted that exploring and evaluating appropriate algorithms for CARS is out of the scope of these experiments and more advanced techniques could obviously be applied. The experimental settings used for the evaluation of DataGenCARS are shown in Table 8.3.

Parameter	**Default Value**
Amount of data for training	70%
Amount of data for testing	30%
Traditional recommendation algorithm	SVD
Classification algorithm for contextual modeling	Naïve Bayes
Recommendation threshold	3

Table 8.3: Experimental settings for the evaluation of DataGenCARS.

We use the implementation of SVD provided by Apache Mahout [Apa14] and the implementation of Naïve Bayes provided by Weka [Uni93]. We apply a *10-fold random subsampling* [DGB09] validation strategy, which implies repeating each experiment k times (k folds) with random samples for the training and testing set; in our case, in each fold we consider 70% of the data for training and 30% of the data for testing. Table 8.4 summarizes the experiments performed and their purpose.

It is important to emphasize that the goal of the experimental evaluation presented in this section is not to show or propose a suitable recommendation algorithm, but just to illustrate the benefits of a synthetic dataset generator and the possibility to incorporate context information to the datasets. Indeed, other recommendation algorithms, different from the ones evaluated, could have been applied.

8.3.1 Set of Experiments 1: Generating and Exploiting a Synthetic Dataset

In this first set of experiments, we focus on a scenario of restaurant recommendations for mobile users located in the state of California. Specifically, we have synthetically generated a number of ratings (this number varies depending on the experiment), which are values in the range from one to five, for a scenario consisting of 943 users, 1682 restaurants, and 900 contexts. Each rating is tagged with a time and date in the range of the years 1980–2000. The schemas of users, types of items, and contexts

Experiment	Sect.	Purpose
Set of Experiments 1: Generating and Exploiting a Synthetic Dataset	8.3.1	Motivate and show the interest of a synthetic data generator like DataGenCARS, by showing how it can generate synthetic data and its usefulness.
Evaluating the Interest of a Context-Aware Approach	8.3.1.1	Show the interest of CARS, and so the need of context-enriched datasets, which can be generated by DataGenCARS.
Evaluating the Impact of the User Uncertainty	8.3.1.2	Show the impact of uncertainty regarding the behavior of users, and so the interest of capabilities to represent uncertainty in a dataset generator like DataGenCARS.
Evaluating the Impact of the Amount of Context	8.3.1.3	Show the impact of the amount of context available, and so the interest of datasets that have rich context information (unlike usual real datasets), that can be generated by DataGenCARS. Show also the interest of capabilities to represent the availability of varying amounts of context data in a dataset generator.
Set of Experiments 2: Generating Realistic Datasets	8.3.2	Show the flexibility of DataGenCARS to generate any required dataset.
Flexible Generation of Datasets of Items	8.3.2.1	Show the flexibility of DataGenCARS to generate any desired dataset of items. In this experiment, the desired features are inspired by real observations available in the Iris dataset.
Flexible Generation of Context-Enriched Recommendation Datasets	8.3.2.2	Show the flexibility of DataGenCARS to generate context-enriched ratings. In this experiment, the desired features are inspired by real observations available in the LDOS-CoMoDa dataset. Show also how to enlarge an existing dataset and how to complete a dataset that has missing values.

Table 8.4: Summary of experiments on DataGenCARS and their purpose.

considered, are defined as follows (in the case of categorical context attributes, we indicate the possible values in brackets):

- Users: age, gender, occupation.
- Restaurants: web_name, address, province, country, phone, weekday_is_open, hour, type_of_food, card, outside, bar, parking, reservation, price, quality_food, quality_service, quality_price, global_rating.
- Contexts: transport_way (walking, bicycle, car, public), mobility (stopped, moving), weekday (week, weekend), mood (happy, sad, active, lazy), season (spring,

summer, autumn, winter), companion (alone, friends, family, girlfriend, children), temperature (warm, hot, cold), weather (sunny, cloudy, rainy, snowing), distance (near, far), time_of_day (morning, night, afternoon).

Unless specified otherwise in the following, we modeled user profiles with six context variables: two corresponding to the context of the user (transport_way and distance) and four corresponding to features of the items (parking, price, quality_food, and quality_service). By default, all the information about the context of the user is available for training and testing, and there is no uncertainty in the utility functions defining the user profiles (i.e., the weight for the extra context attribute *others*, representing unknown factors, is zero). It should be noted that the user utility functions are used only for data generation purposes, but obviously they are assumed to be unknown, and therefore not available to the recommendation algorithms. In Chapter 5, we explain in more detail the context attribute *others* and utility functions.

8.3.1.1 Evaluating the Interest of a Context-Aware Approach

In this experiment, we evaluated the differences between the two recommendation approaches (SVD and CM) for different numbers of ratings. So, four datasets were generated with $10,000$, $50,000$, $100,000$ and $200,000$ ratings, respectively. The results comparing the mean squared errors (MAE) for the ratings predicted and the F-measure metric (F1 with $\beta = 1$), that combines the effects of the recall and precision, are shown in Figure 8.16. As we can see, an approach that takes into account the context factors achieves better results; the only exception observed is in the case of the MAE for a low number of ratings ($10,000$). Moreover, the model improves as the number of ratings available to learn the model increases. The positive effect of a higher number of ratings also holds with a traditional approach, but the impact is more moderate in that case.

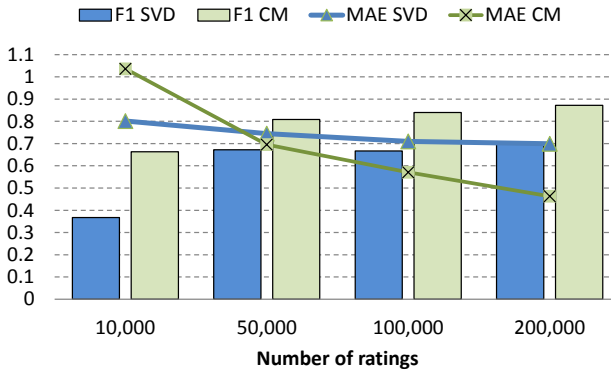


Figure 8.16: SVD vs. CM (Naïve Bayes): two classes (recommend / not recommend).

A similar trend can be observed if we consider the problem as a multi-class classification problem where each potential rating from one to five is a class, as shown in

Figure 8.17. Nevertheless, the effect of having five classes instead of only two leads to an overall decrease in the performance, as expected.

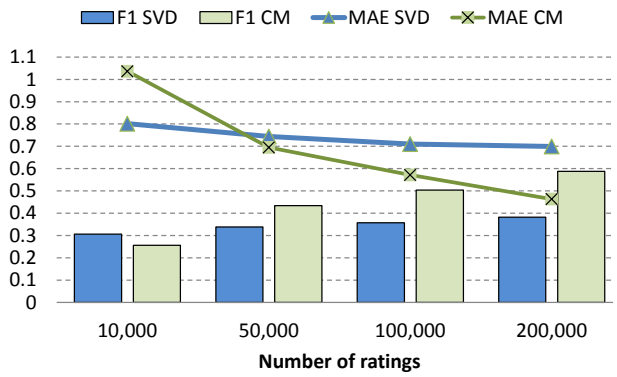


Figure 8.17: SVD vs. CM (Naïve Bayes): five classes (integer value of rating).

This experiment shows that taking into account context data can have an impact on the recommendations. So, it motivates the interest of having datasets including context information, as DataGenCARS can provide.

8.3.1.2 Evaluating the Impact of the User Uncertainty

In this experiment, we evaluated the difference in the behavior of the two approaches for different degrees of uncertainty regarding the rating behavior of the users and considering $50,000$ ratings. To achieve this goal, we defined user profiles with a weight different from zero for the special context attribute that represents factors not modeled but that have an influence on the rating behavior of the user. For example, an extreme uncertainty percentage of 100% (i.e., a weight of 1 for the special attribute "others" in the user profiles and a weight of 0 for all the remaining attributes) represents a user that rates items in a way apparently random, as the actual rating may be based on attributes that have not been modeled in the user profiles. Introducing this uncertainty leads to difficulties to learn appropriate models for the users, as it implies that there are unknown attributes that have an impact on the behavior of the users and whose values are not available.

Figure 8.18 shows how increasing the uncertainty leads to a higher MAE and a smaller F-measure. The CM approach performs better in terms of precision, recall, and F-measure. However, the traditional approach SVD outperforms CM in terms of the absolute predicted rating (MAE) when there is significant uncertainty; nevertheless, as shown in Figure 8.16, the MAE decreases more abruptly with the number of ratings in the case of CM, so this effect is minimized when we increase the number of ratings over $50,000$.

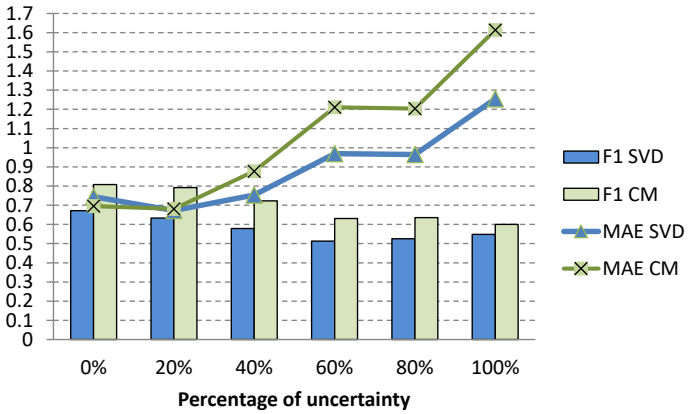


Figure 8.18: Impact of the User Uncertainty.

8.3.1.3 Evaluating the Impact of the Amount of Context

In this experiment, we evaluate the impact of the amount of context information available on the performance of the context-aware recommendation algorithm considered. For this purpose, we generate seven datasets of 50,000 ratings each, but with a different average percentage of context information available, from 100% (i.e., all the context data were provided with the ratings) to 10% (i.e., on average 10 out of each 100−sized chunk of values of the context variables are available). Figure 8.19 shows how the MAE increases and the F-measure decreases when a higher portion of the context data is unknown. For easier interpretation, we also show on the right part of the figure the results obtained by a *random* recommendation approach (that randomly predicts a rating between one and five for each element in the testing set) and those obtained by a *baseline* recommendation approach that simply predicts the most frequent rating.

8.3.2 Set of Experiments 2: Generating Realistic Datasets

In this set of experiments, our goal is to show how DataGenCARS can help in the generation of realistic datasets. First, in Section 8.3.2.1, we present an experiment where we generate a synthetic dataset of items that exhibits features similar to those present in a pre-existing real dataset. Afterwards, in Section 8.3.2.2, we consider an existing context-based dataset used for the evaluation of context-aware recommendation systems and we use it as a basis to generate synthetic datasets.

8.3.2.1 Flexible Generation of Datasets of Items

In this experiment, we evaluate the flexibility of DataGenCARS to generate any desired dataset by specifying the features required for the data to generate, which could be based on situations observed in the real world. For simplicity, we focus on

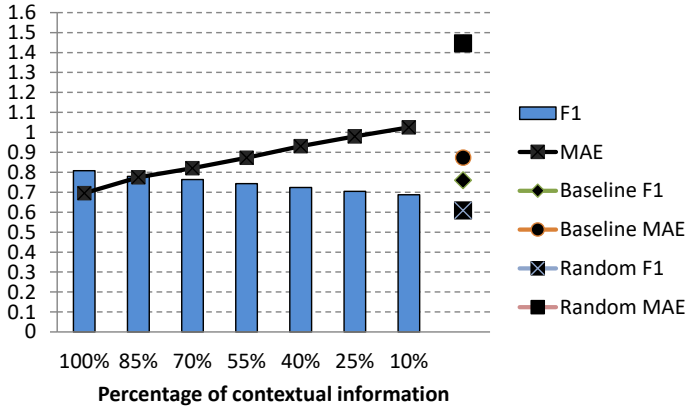


Figure 8.19: Impact of the amount of context available.

the generation of a dataset of items. For practical reasons, we extracted the desired features of the synthetic dataset to generate from an existing dataset: the popular Iris plants dataset [Fis88], which is one of the example datasets provided by Weka.

First, we analyzed the main features of the Iris dataset and we observed that there is a strong correlation between some of its attributes. This implies that some attributes have to be generated at the same time to ensure consistent values. For that purpose, we defined an item profile for each of the three classes considered in the Iris dataset. The attributes considered relevant for the item profile are in this case "petallength" and "petalwidth", and for each of the three classes we determined the appropriate range of values for those attributes, based on the real dataset available. Then, we generated 150 instances (like in the original dataset) and we evaluated a Naïve Bayes classification algorithm on both the original and the synthetic datasets.

In Figure 8.5, we show the performance metrics obtained with the original and synthetic datasets, by using Weka. As it can be observed, the performance over the two datasets is similar.

	MAE	**Precision**	**Recall**	**F-measure**
Original dataset	0.0342	0.96	0.96	0.96
Synthetic dataset	0.0177	0.987	0.987	0.987

Table 8.5: Performance metrics obtained with the original and synthetic datasets (Iris).

Moreover, we also evaluated the performance of the classification algorithm when the entire synthetic dataset is used for training and the entire real dataset is used for testing, obtaining the following performance metrics: $MAE = 0.0519$, $precision = 0.949$, $recall = 0.94$, and $F\text{-}measure = 0.94$, which are also similar to the values obtained in Table 8.5. We also compared the attribute values themselves (average

values, standard deviation, percentage of unique values, and number of items of each profile), and the results obtained are also similar. Finally, we also performed other tests generating more instances (500 and 1000), which lead to similar conclusions.

It should be noted that the purpose of this experiment is simply to show the flexibility of DataGenCARS to generate any dataset required, by specifying the features of the data that must be generated. In this case, we extracted the features desired from an existing dataset just to be able to easily evaluate the suitability of the dataset synthetically generated. Obviously, if we already have a dataset with the features required, the original dataset can be directly used and there is no need to generate a synthetic one. Nevertheless, in some cases it might be interesting to extract the features from an existing dataset, change some features (statistics) as desired, and generate a new dataset similar to the initial one but with modifications in some of its original features: this would allow assessing the impact of these changes on the performance of the recommendation algorithms evaluated. As an example, it could be interesting to take an existing dataset, add temporal information (e.g., timestamps) to the ratings, segment those ratings according to specific time periods, and adjust the ratings to simulate different rating behaviors of the users along time; in this way, we could evaluate how a recommendation algorithm would perform when the ratings are subject to seasonality (e.g., movie ratings for new releases may exhibit different patterns than ratings for older movies, some items are more popular during certain periods of the year, some preferences may change along time, etc.), even if the original dataset does not include temporal information. As another example, it could be interesting to evaluate the performance of a recommendation algorithm when there are users behaving unexpectedly (e.g., submitting random ratings) and/or submitting highly-biased and unreliable ratings; with DataGenCARS we could easily simulate users that behave as spammers or that show a random behavior. In some cases, we could be interested in removing redundancy or repeated user ratings in the original dataset (e.g., the same user evaluating a single item several times in the same context). As a final example, we could be interested in removing attributes from the existing dataset or complete missing data, to show their impact on the performance of the recommendation algorithm that must be evaluated.

8.3.2.2 Flexible Generation of Context-Enriched Recommendation Datasets

Now we consider a real dataset that incorporates context information and that can be used for the evaluation of context-aware recommendation algorithms. The goal of the experiments presented in this section is to show that DataGenCARS can generate suitable synthetic datasets that can mimic the desired characteristics of existing scenarios. For that purpose, we consider, as a basis for comparison and inspiration, the LDOS-CoMoDa dataset [KOK+11, Košl1, OTTK13], which is a movie recommendation dataset that incorporates context information. Specifically, it contains the following information:

- *User*: age, sex (male, female), city, and country.

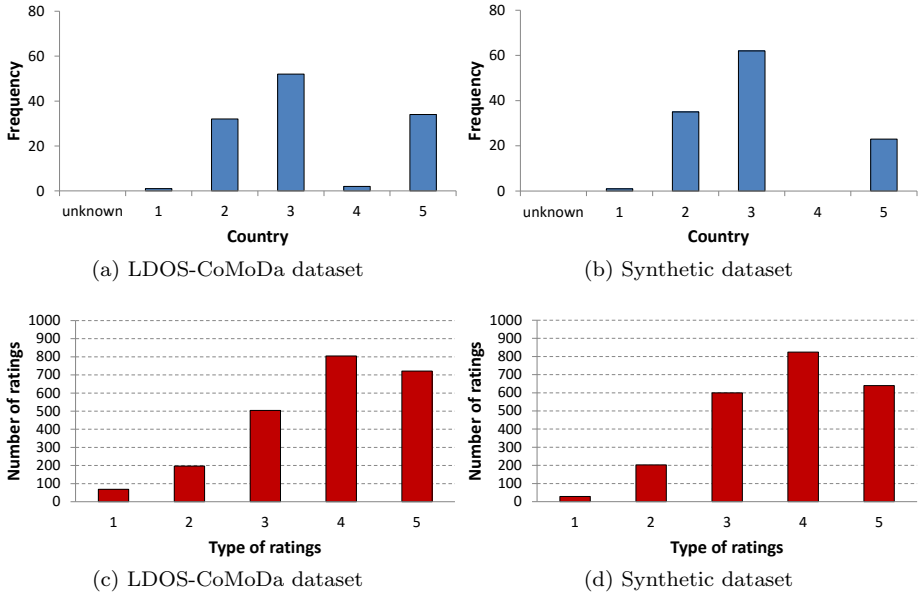
- *Movie*: director, country, language, year, genres, actors, and budget.
- *Context*: time (morning, afternoon, evening, night), day type (working day, weekend, holiday), season (spring, summer, autumn, winter), location (home, public place, friend's house), weather (sunny, rainy, stormy, snowy, cloudy), social environment (alone, partner, friends, colleagues, parents, public, family), end emotion (sad, happy, scared, surprised, angry, disgusted, neutral), dominant emotion (sad, happy, scared, surprised, angry, disgusted, neutral), mood (positive, neutral, negative), physical status (healthy, ill), decision point (user decided which movie to watch, user was given a movie), and interaction status for movies that are rated more than once by the user (first interaction with a movie, n-th interaction with a movie).
- *Rating*: on a scale of one to five.

Although the LDOS-CoMoDa dataset is not freely available, it is possible to request it by contacting the researchers that own it at the University of Ljubljana (User-adapted Communications & Ambient Intelligence Lab). As in the experiment shown in Section 8.3.2.1 (Iris dataset), we extracted the features desired from an existing dataset just to be able to easily evaluate the suitability of the dataset synthetically generated. For these experiments, we decided not to use the STS dataset (studied in Section 8.2.1) because the context attributes in the STS dataset contain many unknown values. The LDOS-CoMoDa dataset also lacks contextual information but to a lesser degree.

Generating a Similar Dataset

The first experiment generates a synthetic dataset that tries to replicate the original LDOS-CoMoDa dataset, by applying the workflow described in Figure 5.10 (see Chapter 5). After replicating the dataset, we compared the histograms of the different user attributes in the original and generated datasets, the distributions of the context variables in both, and the corresponding statistical properties of the ratings generated. For illustration purposes, we show some histograms in Figure 8.20, where for simplicity we represent the values of qualitative attributes with numbers rather than showing the corresponding label. The similarities can also be observed for other attributes not shown in the figure. We have compared the histograms of all the individual context attributes in both the original and the synthetic dataset. The average difference in the mean values of the variables is 0.022 and the average difference in the variance is 0.034. Moreover, the maximum difference in the mean is 0.074 (for an attribute representing the *dominant emotion*, that can take seven different values) and the maximum difference in the variance is 0.146 (for an attribute representing the *social environment*, that can take seven different values). The average correlation coefficient when comparing the frequency distribution of values of each original and synthetic variable is 0.9989, being the minimum value 0.9934 (for the attribute representing the *dominant emotion*), which indicates a very strong positive correlation between the original dataset and the generated one.

The results show the ability of DataGenCARS to generate datasets of appropriate quality, including the capability to reproduce context data perceived in available real-world situations. It should be noted that, instead of considering a real dataset (the LDOS-CoMoDa dataset, in this case), we could have performed this experiment by comparing two datasets generated by DataGenCARS; however, the results obtained in that case would be optimistic, as we would be using the same tool to generate the two datasets.



(a) LDOS-CoMoDa dataset

(b) Synthetic dataset

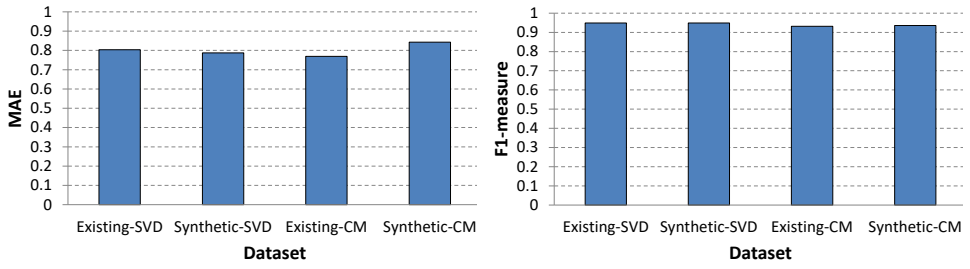
(c) LDOS-CoMoDa dataset

(d) Synthetic dataset

Figure 8.20: Generating a dataset similar to LDOS-CoMoDa: some example histograms.

In Figure 8.21, we show the results of the recommendation algorithms for the original LDOS-CoMoDa dataset and the synthetic dataset generated based on LDOS-CoMoDa. The algorithm applied when context variables are considered is CM and the algorithm applied when context variables are not taken into account is SVD (like in Section 8.3.1). As we can observe in the figure, the results are similar in both cases, which shows that the synthetic dataset is able to capture the main features of the original existing dataset. By comparing the performance on both the original and the synthetic datasets, we can observe that the maximum difference in the MAE is 0.073 and the maximum difference in the F1-measure is 0.017. These differences can be considered to be very small: the algorithms predict ratings between 1 and 5 and the MAE is less than 0.1 in the worst case, which represents a very small deviation of the accuracy.

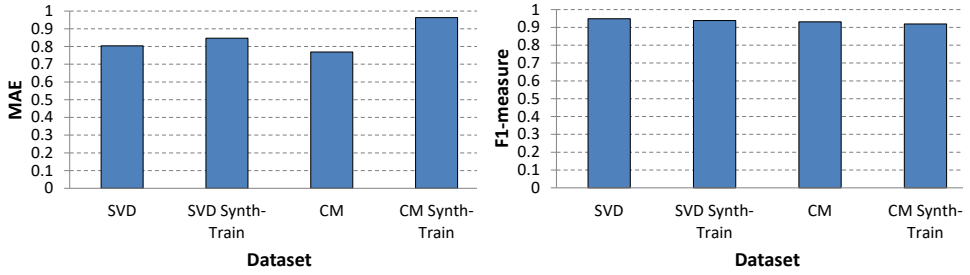
An alternative way to evaluate the differences observed when the dataset is gener-



(a) MAE (b) F1 measure

Figure 8.21: Synthetic dataset based on an existing one (LDOS-CoMoDa): recommendation performance (considering the datasets separately).

ated synthetically is shown in Figure 8.22. In this case, we compare the performance of the recommendation algorithm with the original dataset (in each fold, 70% of the data is used for training and 30% of the data is used for testing) with the performance obtained when the recommendation algorithm is trained using the entire synthetic dataset and the entire real dataset is used for testing. Again, the differences observed are quite small.



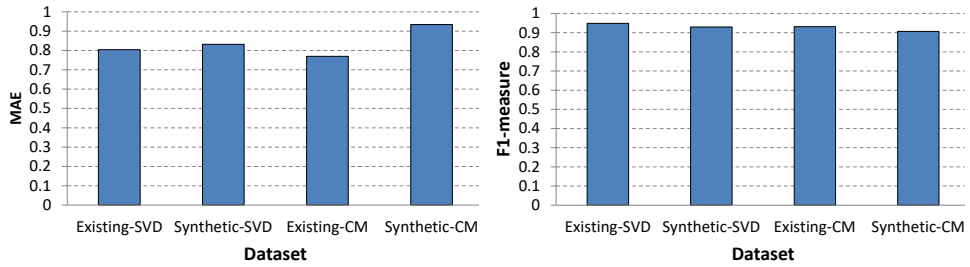
(a) MAE (b) F1 measure

Figure 8.22: Synthetic dataset based on an existing one (LDOS-CoMoDa): recommendation performance (only the original dataset vs. training with the synthetic dataset).

Enlarging a Dataset

To generate the results shown in Figure 8.23, we follow a different strategy. We select a 10% of the ratings of each user in the original dataset LDOS-CoMoDa and we apply the workflow described in Section 5.2.3 to generate a higher number of ratings (the number of ratings that were available in the whole original dataset LDOS-CoMoDa, that is, 2296 ratings). Again, the results obtained with the original and synthetic

datasets are similar.

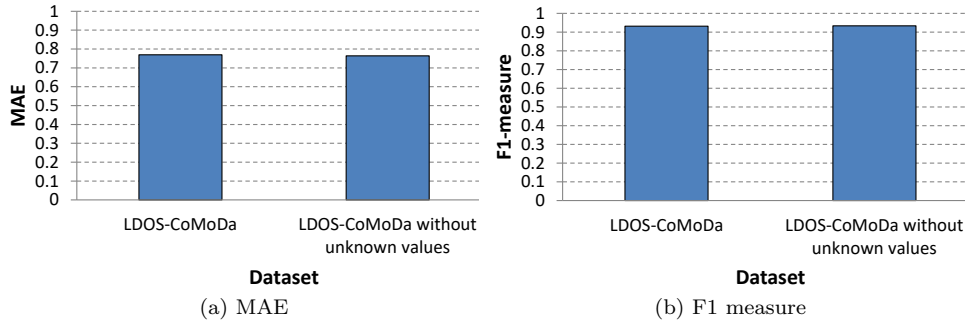


(a) MAE (b) F1 measure

Figure 8.23: Synthetic dataset based on a subset of an existing one (LDOS-CoMoDa): recommendation performance.

Completing a Dataset

As a final experiment, we generate a new dataset by replacing the unknown values in the original dataset LDOS-CoMoDa with other values (i.e., we replace unknowns, as described in the workflow presented in Section 5.2.4). Figure 8.24 shows the results obtained when applying the recommendation algorithm CM to both datasets. The results obtained are similar, as in both cases we are using a recommendation algorithm that takes into account the context variables and the percentage of unknown values in the original dataset is, in this case, only 3.27%.



(a) MAE (b) F1 measure

Figure 8.24: Completion of unknown values: recommendation performance.

8.4 Experimental Evaluation of a Use Case Scenario

In this section, we present an experimental evaluation considering a specific example scenario: the recommendation of works of art in the MoMA museum (see Chapter 6). This evaluation has been performed to show the feasibility and the benefits that context-aware mobile recommendations can offer over other recommendation strategies. First, we describe the experimental settings used for the evaluation of a mobile P2P context-aware recommendation approach. Then, we discuss the experimental results obtained.

8.4.1 Experimental Settings for the Museum Scenario

The scenario of the museum and the simulation prototype developed were described in detail in Chapter 6. In these experiments, the evaluated user is initially located at the front door of the museum and follows a path suggested by the context-aware recommendation application installed on her/his mobile device, while the other museum visitors start from a random location within the museum and follow other synthetically-generated trajectories. The experimental settings used for evaluation are shown in Table 8.6.

In the experiments performed, we evaluate the *Trajectory and User-Based Collaborative Filtering* approach described in Section 4.3, that we will denote T&UBCF, by using pure mobile P2P ad hoc communications. Rather than assuming the availability of a centralized server storing a large database of rating information, the idea is that mobile users will propagate partial amounts of rating data in an opportunistic way, that is, when they meet each other in the physical space. As baselines for comparison, we also consider the following recommendation alternatives:

- *Completely-random recommender* (FULLY-RAND): the user visits works of art recommended in a completely random manner, even if this means changing from one room to another that may be located far away. This is expected to be the worst approach possible, as the user could potentially have to traverse very large distances between observations.

- *Exhaustive visit recommender* (ALL): the user is recommended to visit all the works of art in her/his current room, then an exit (the stairs, that allow changing from one floor to another, or a door) is recommended randomly, guiding the user to a different room, and so on.

- *Nearest Point Of Interest recommender* (NPOI): the user is recommended to go to the nearest POI (a work of art or an exit) in the museum. If the nearest POI is an exit/door, then the user leaves the current room.

- *Centralized recommender* (*Centralized*) [dCRHIHTL17c]: a central server stores information about all the ratings that are released along time and, based on all these data available, a user-based collaborative filtering strategy is applied. The

Parameter	Default value
Number of items	240 items
Number of simultaneous visitors (floors 4 and 5)	176 visitors (inspired by [PdSS16])
Trajectories followed by other visitors in the museum	50% of the visitors moving always to the nearest POI (NPOI strategy) & 50% of the visitors observing works exhaustively (ALL strategy)
Visiting time in the museum	1 hour
Visitor's average speed	3 Km/h
Observation time (of a painting or sculpture)	30 seconds
Time needed to change to another floor (take the stairs or the elevator)	60 seconds
Number of recommended items to keep in the result list (K)	10 items
Recommendation threshold (1-5)	2.5
Knowledge base increase threshold	40 new ratings
Minimum time interval between successive recommendation updates	30 seconds
Similarity threshold for the UBCF algorithm	0.5 (Pearson correlation)
TTL of the data to propagate	3 minutes
Communication latency	1 second
Communication bandwidth	54 Mbps (IEEE 802.11g)
Communication range	250 meters
Communication obstacles	Walls block signals (each data communication limited to a single room)
Retransmission period	1 second

Table 8.6: Experimental settings for the museum use case scenario.

k items with the highest estimated rating and exceeding the predefined recommendation threshold are re-ordered in order to minimize the distance traversed by the user, and the sorted list is finally recommended to the user.

- *Know-It-All recommender* (Know-It-All): the trajectory and user-based collaborative filtering strategy T&UBCF is applied, but assuming complete knowledge about all the real ratings that the other visitors would provide.
- *K-Ideal recommender* (K-Ideal): the real ratings are considered and the k items with the best real ratings (not seen yet by the user) are recommended, after re-ordering them according to the shortest trajectory passing through those items.

All the strategies were implemented within the context of the generic framework described in Chapter 3 and using Apache Mahout [Apa14]. For the strategies based on collaborative filtering, in order to alleviate the cold start problem, when there is not enough data to obtain a recommendation we apply a default NPOI strategy.

The last two strategies assume complete knowledge about the real ratings that the other visitors would provide for each work of art in the museum. Obviously, it is unrealistic to assume that this information could be available (it includes knowledge about votes that have not been released), and therefore the performance of these last two approaches could be considered as a top-level performance that might be potentially obtained if complete knowledge about the other visitors was available.

8.4.2 Experimental Results for the Museum Use Case

In this section, we present the experiments performed and the results obtained, that show the interest of our proposal. In this set of experiments, we evaluate a *P2P* deployment of our T&UBCF recommendation approach. The main goal of these experiments is to analyze the potential feasibility of deploying mobile CARS by using exclusively mobile P2P communications. Rather than assuming the availability of a centralized server storing a large database of rating information, the idea is that mobile users will propagate partial amounts of rating data in an opportunistic way, that is, when they meet each other in the physical space.

In Figure 8.25, we show the average of the ratings provided by the user for the items she/he observed in the museum, considering a rating scale in the range of one to five. For the mobile P2P approach, we considered two possible cases: one where the TTL of the data transmitted is set to three minutes (i.e., a vote stops propagating after three minutes of its initial release) and another one where the TTL is ∞ (i.e., all the data are kept alive in the network during the whole simulation). In both cases, we observe that the P2P strategy achieves a higher average rating than most strategies (FULLY-RAND, NPOI, and ALL). The only exceptions are the *Centralized* strategy, that behaves slightly better than mobile P2P thanks to a higher availability of information (all the data are collected by a centralized server and made available to the recommendation process), and Know-It-All and K-Ideal, which are unrealistic ideal alternatives. Although the differences between the ALL and P2P strategies do not seem very significant, the ALL strategy implies that the user sees more items that he/she does not particularly like (*no likes*), as we will explain below with Figure 8.26.

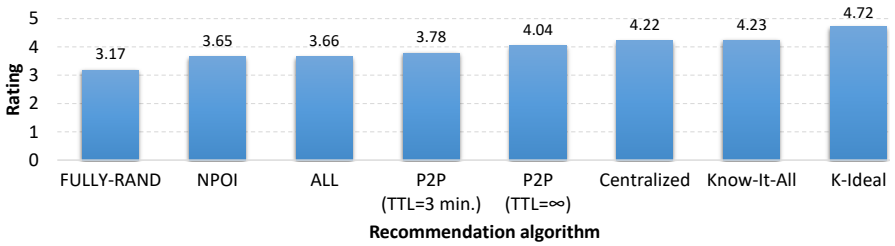


Figure 8.25: Average rating provided for the items observed in the museum use case.

Figure 8.26 shows the number of *likes* (items observed and particularly-well rated by the user with a score no smaller than 3.5 in a scale of one to five), *no likes*

(items rated by the user with a score below 3.5), and the difference between them. The P2P variants exhibit a performance close to the centralized strategy and are only outperformed by that centralized approach and by the ideal and unrealistic strategies. Although NPOI achieves a high number of likes, it also leads to a considerably high number of no likes, and so an overall higher dissatisfaction. Besides, as it was shown in Figure 8.25, the average rating obtained with NPOI is also lower than with other strategies.

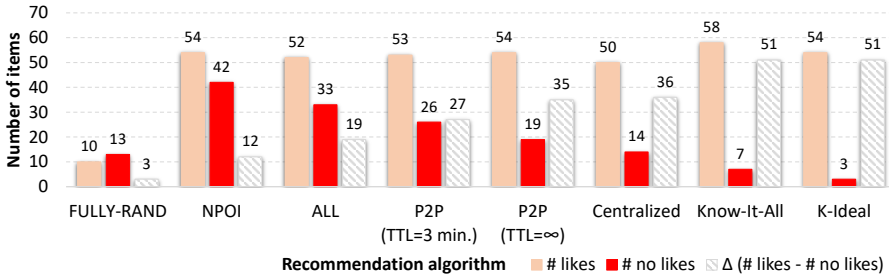


Figure 8.26: Likes and no likes in the museum use case.

We also analyzed the evolution along time of the proposed mobile P2P recommendation approach. First, in Figure 8.27 we can see the number of votes released along time. Figure 8.28 depicts the *MAE* in the predicted rating of the items observed by the user during her/his visit to the museum. The dashed trend line depicted in Figure 8.28 shows that, as expected, the quality of the recommendations improves along time (i.e., the MAE decreases), as the knowledge base stored on the user's mobile device (initially empty) increases in size thanks to the mobile P2P communications with other users.

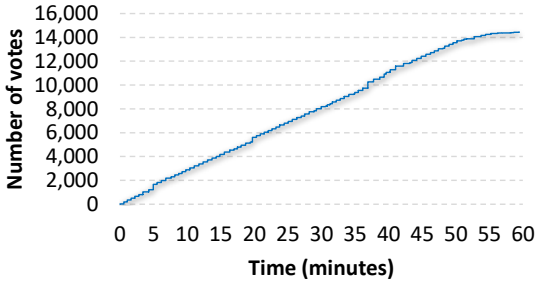


Figure 8.27: Votes provided by the visitors along time.

Moreover, we have also evaluated the impact of other intermediate TTL values (between 3 minutes and ∞) on the performance of the mobile P2P recommendation approach (see Figures 8.29 and 8.30): higher TTL values eventually lead to larger local databases and a higher recommendation accuracy, at the expense of a higher number of network communications.

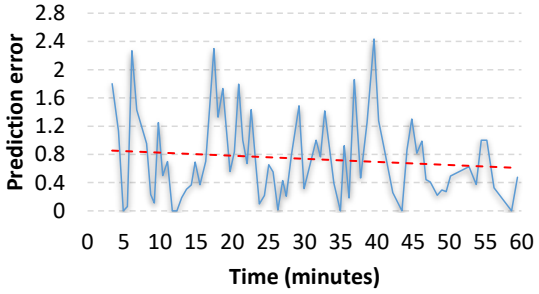
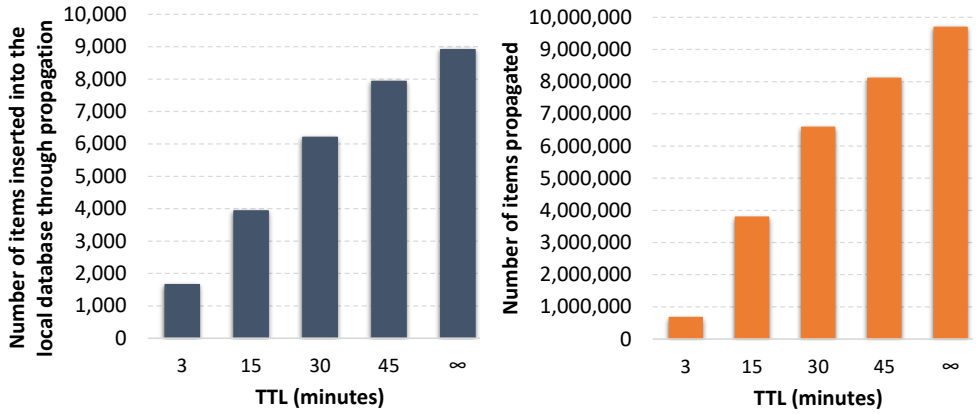


Figure 8.28: Prediction error along time.



(a) Number of items stored in the local database.

(b) Number of items propagated.

Figure 8.29: Item propagation for different TTL values in the museum use case.

As a conclusion, the experimental results show that the mobile P2P recommendation approach provides satisfactory results while avoiding the potential inconveniences of a centralized approach. Obviously, the centralized approach is slightly better because it applies the same recommendation process but it exploits a larger database to learn appropriate recommendation models; instead, the mobile P2P alternative relies on a local database with information collected opportunistically by the mobile device, rather than a centralized database with all the information generated until then.

The experiments presented above are a set of representative results. Through other experiments using user trajectories that lead to highly-skewed distributions of visitors in the museum, we have observed that in those cases the cold start problem might re-appear when there is a major change in the user's location: as the information propagated through the ad hoc network tends to spread over the spatial area only gradually, reaching an area with a small number of visitors could be problematic at the beginning; for example, if the user moves to a floor of the museum where the number

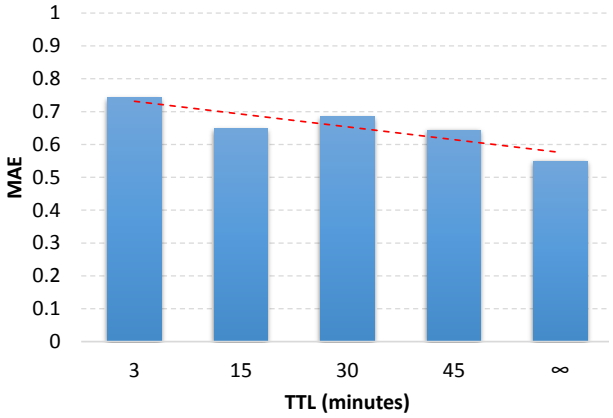


Figure 8.30: MAE for different TTL values in the museum use case.

of visitors is very small, it could happen that the local database on her/his mobile device initially has little information about the works of art in that floor, which would lead to poor predictions about those works. A mixed P2P approach, based on some support fixed nodes located at strategic places and storing some information about ratings released in nearby areas, could be used to help to diminish this problem.

8.5 Summary of the Chapter

In this chapter, we presented a prototype of the MOONRISE framework and a set of experiments that show the interests of our MOONRISE framework and the DataGenCARS synthetic dataset generator. In our proposal, we first compared the different pull-based recommendation paradigms (pre-filtering, post-filtering, and contextual modeling) for mobile environments. Then, we presented an evaluation of DataGenCARS. A complete experimental evaluation showed the interest of a tool like this one, given the scarce availability of rich context-enhanced datasets that can be used for recommendation evaluation purposes. Finally, we described the experimental evaluation that we have performed in a real use-case scenario of recommendations of works of arts to observe in the MoMA museum. The results obtained show the interest of context-aware mobile recommendations. Besides, this case study also represents a contribution to fill the gap between the design and evaluation in CARS, as we exploited the DataGenCARS tool in a mixed scenario built using both real data and synthetic data, to evaluate a recommendation approach for an indoors scenario.

In addition to the experiments presented in this chapter, in Appendix C, we present experimental results for two alternative solutions to the problem of identification of the type of item that the user specifies in a pull-based recommendation: one based on the use of the Hidden Markov Model and another one exploiting Information Retrieval techniques (described in Section 4.1.6).

Chapter 9

Conclusions

In this chapter, we present some conclusions about the work presented in this dissertation. First, we summarize our main contributions to the field of context-aware recommendation systems. Second, we discuss the evaluation results obtained. Finally, we indicate some open lines of work for future research.

In this thesis, we have focused on the study of mobile context-aware recommendation systems and presented a novel context-aware recommendation architecture that aims to facilitate the development of context-aware recommendation applications for mobile environments. It supports up-to-date recommendations of relevant items to users with mobile devices moving through predefined paths (e.g., road networks) or freely in the space, different scenarios of recommendation, static and dynamic context information, pure P2P ad hoc communications, as well as recommendations based on pull and push approaches. In this way, we comply with our general research goal and answer the scientific questions initially identified during the research design. We briefly summarize in the following those research questions and the tasks performed in this thesis to answer them:

- *What is the technological background related to context-aware recommendations in mobile environments?*

  - We studied the current state of the art of context-aware recommendation systems for mobile users. During the research process, we paid particular attention to the fact that, in addition to other context information, the mobility of users and items is important to determine the appropriate answer to a recommendation query or to proactively suggest items of interest, and proposed to modify the context-aware recommendation paradigms to keep the answer up-to-date. With our research, we attempt to overcome the existing gap between context-aware recommendation systems and mobile computing.
  - We analyzed the theoretical bases related to the fields of context-aware

recommendation systems and mobile computing, with the purpose of improving the effectiveness of the recommendations.

- *How could we develop data management techniques that enable the deployment of a context-aware recommendation system for mobile environments?*
  - We designed a context-aware recommendation architecture for mobile computing environments, that can facilitate the development of context-aware recommendation applications for mobile users.
- *How could we determine the potential interest of these types of context-aware mobile recommendations?*
  - We performed a set of experimental evaluations that show that the effectiveness of context-aware recommendations for mobile users can improve by considering contextual information in the recommendation process.
- *How could we evaluate the architecture proposed for the validation of the results in several application domains?*
  - We developed an automatic generator of synthetic datasets, called DataGenCARS, in order to alleviate the problem of the lack of datasets suitable for the evaluation of context-aware recommendation systems.
  - We developed a simulation application for a specific use case scenario, due to the difficulty of testing our architecture in a real environment. It supports the simulation of a museum scenario to test context-aware recommendation systems for mobile users that may exchange data by using pure P2P ad hoc communications. We performed several experimental evaluations that show the benefits of our proposal.

As far as we know, *no existing work addresses the development of flexible and generic architectures that facilitate the implementation of context-aware recommendation systems for mobile computing environments over a distributed infrastructure*. Furthermore, we are also pioneers in implementing a synthetic data generator for the evaluation of context-aware recommendation systems, called DataGenCARS, motivated by the lack of datasets available for that purpose. Both contributions are a relevant novelty in the literature of context-aware recommendation systems and mobile computing.

9.1 Main Contributions

In the following, we describe in more detail the main contributions of our work.

9.1.1 Context-Aware Mobile Recommendation Architecture

Our proposed architecture is generic, extensible, and adaptable to the requirements of specific types of recommendations. It is composed of *three architectural layers*, that contain the necessary modules to exploit context-aware recommendations in mobile environments.

The architecture provides several *traditional recommendation algorithms* (e.g., collaborative filtering recommendation and content-based recommendation algorithms), the possibility to *combine several recommendation algorithms*, and *context-aware recommendations* based on pull and push approaches. As the pull-based recommendation and push-based recommendation approaches are designed for mobile environments, dynamic context information can be exploited, which can be obtained by using *sensors* embedded in the mobile devices of the users. The modules access the local database of the mobile user, by using the *repository manager* module, for the generation of recommendations of items, which are automatically updated as needed (e.g., at a certain refreshment frequency).

Moreover, the information exploited by context-aware recommendation systems, can be stored in a distributed way on the users' mobile devices, that can disseminate their local data by using pure P2P networks that exploit exclusively short-range wireless ad hoc communications (e.g., Wi-Fi, Bluetooth, etc.). In this scenario, the mobile devices of the users would behave as peers and enrich their local database through *data sharing* (exchange of rating information) between them under specific conditions and in an opportunistic way, when they become neighbors from a communication point of view.

9.1.2 Context-Aware Mobile Recommendation Approaches

In the logic layer of the architecture, the *pull-based recommendation* and *push-based recommendation* approaches are the main modules. Both approaches are generic and can be adapted to different mobile computing scenarios and domains. In this way, with the proposed architecture, recommendation systems could *dynamically capture and update context information* without the user intervention, by using different *sensors* embedded in the mobile devices. Thus, mobile users could minimize the effort of explicitly entering static context information into the recommendation system. Furthermore, they could *automatically update the list of recommended items* (e.g., at a certain refreshment frequency).

In addition, these recommendation approaches allow the user to optionally specify her/his *context criteria* (or *preferences*) about the importance of different context variables, by adjusting one or all the weights of the variables. For example, for a user who requires restaurant recommendations, the most relevant context variable could be the transport way she/he is using and the distance to each restaurant. Finally, they can apply *hard and soft context constraints*. Thus, the mobile user can receive items that satisfy specific conditions (on the values of certain context variables), by specifying hard constraints (e.g., a location-dependent constraint that requires that the recommended items must be near the user), or retrieve items according to

her/his preferences (regarding the impact of the different context variables), which are understood as soft constraints.

On the one hand, *pull-based recommendations* provide relevant items to the mobile user based on an explicit query submitted by her/him. The main features of this approach are:

- It supports the specification of an *item type* (e.g., restaurant) or *keywords* (e.g., place to eat) as the query. In case the user enters keywords, the system is able to infer the item type required by the user, by using HMM-based and IR-based methods. In this way, users can specify queries that fit their needs and without having to know in advance the item types that have been defined in the recommendation system.

- It evaluates the user's request as a *continuous query*, and thus the items suggested to the user are automatically updated by the recommendation system (until the query is explicitly canceled by the user), recalculating the answer to the corresponding query in an efficient manner when the contextual situation changes.

- It allows applying any of the *pre-filtering*, *post-filtering* and *contextual modeling* recommendation paradigms, which incorporate context information in the recommendation process in order to increase the effectiveness of the recommendations.

On the other hand, *push-based recommendations* proactively provide relevant items to the mobile user when the current context situation is considered appropriate. Thus, the recommendation system relieves the user from having to type or introduce significant information as an input. The main features of this approach are:

- It recommends relevant items to the mobile user without an explicit request by her/him.

- It decides when the contextual situation is appropriate to push recommendations to the user, avoiding an overload of information on the user's mobile device.

- It is based on the pre-filtering and post-filtering recommendation paradigms. Thus, in a first phase of pre-filtering, the system ignores items out of the scope of the user's context, and then a traditional recommendation algorithm is used to obtain candidate items to suggest. In a second phase, the post-filtering approach is applied to resolve potential conflicts between candidate items (e.g., recommendations for different overlapped *environments*).

Besides, we considered, as an example of push-based recommendation, a *trajectory-based recommendation* approach, that takes into account context data such as the location of the mobile user and her/his trajectory to proactively push personalized recommendations in real-time.

9.1.3 DataGenCARS: A Synthetic Dataset Generator

One of the major issues that we have identified in the field of context-aware recommendation systems is the lack of suitable datasets that incorporate context information. Motivated by this, we developed DataGenCARS, a synthetic dataset generator that can be used for the automatic generation of datasets which are appropriate for the evaluation of context-aware recommendations algorithms ($User \times Items \times Contexts \rightarrow Rating$). The main features of this tool are:

- It is generic, as it can fit different application domains and sets of needs, by appropriately defining a set of input data files that will direct its behavior.
- It presents a flexible definition of input data files (e.g., user schemas, user profiles, item schemas, and context schemas) for the generation of synthetic datasets.
- It facilitates a realistic generation of ratings and attributes of items.
- It allows mixing real and synthetic datasets.
- It provides functionalities to analyze existing datasets as a basis for synthetic data generation.
- It supports the automatic mapping between item schemas and Java classes.
- It allows applying different workflows, such as the generation of a dataset similar to an existing one, the generation of a completely-synthetic dataset, increasing the number of ratings in an existing dataset, completing unknown context information in an existing dataset, and composing different workflows. The application of one workflow or another depends on the purpose of the dataset to be generated.

A set of experimental evaluations show the interest of a tool like this one, given the scarce availability of rich context-enhanced datasets that can be used for recommendation evaluation purposes. It also shows that it is possible to generate datasets that exhibit any required behavior, including both those observed in real situations and others that could arise in hypothetical scenarios. Therefore, it represents an interesting tool that can help to alleviate the problem of scarcity of datasets suitable for the evaluation of recommendation strategies, especially those that require rich context information.

9.1.4 Experimental Evaluation

Our contributions have been tested through an extensive set of experimental evaluations that, among other aspects, show that:

- Incorporating context data in the recommendation process can improve the effectiveness of recommendation systems for mobile users.

- Considering data distributed among the mobile devices of the users and shared opportunistically in a P2P way is a feasible option.

Besides, we have also performed other experiments, among which we can highlight those that support the development of the DataGenCARS tool.

9.2 Evaluation of Results

The results of the thesis presented in this document have been published in relevant international journals, conferences, and workshops. In the following, we briefly describe these publications in approximate chronological order and grouped by topic:

- *Publications related to the proposed architecture*:
  - In [dCRHI14c], we outlined the basics of our architecture, designed to support an easy development of context-aware recommendation systems for mobile computing environments.
  - In [HITdCRH15], we focused on the design of a theoretical framework that puts forward a novel approach to model push-based recommendation processes for mobile users through the use of the concept of environments.
  - In [dCRHITLG16], we formalized the problem of keyword-based searching for pull-based recommendations and evaluated two alternative solutions: one based on the use of the Hidden Markov Model and another one exploiting Information Retrieval techniques.
  - In [dCRHI16], we described in detail our architecture, that facilitates the development of context-aware recommendation systems for mobile users, focusing particularly on pull-based recommendations, which accommodates the pre-filtering, post-filtering, and contextual modeling paradigms.
  - In [dCRHIHTL17c], we presented a simulation application that allows testing a trajectory-based context-aware recommendation system able to proactively push relevant items to mobile users, assuming the availability of a centralized server that stores a large database about all the ratings that are released along time.
  - In [dCRHITH17], we focused on analyzing the possibility of using pure mobile P2P networks to exchange relevant data in contexts where no centralized database or server exists.
- *Publications related to the synthetic data generator DataGenCARS*:
  - In [dCRHIHTL17a], we presented DataGenCARS, a complete Java-based synthetic dataset generator that can be used to evaluate the suitability of context-aware recommendation systems.
- *Publications related to bibliographic reviews*:

- In [dCRHITLH15], we provided a survey of location-aware recommendation systems in mobile computing scenarios.
- In [IHTLdCRH15], we presented a review of the state of the art on the use of sensors in mobile context-aware recommendation scenarios.

During the thesis we have also presented our research plans and some intermediate results in national conferences and events, such as [dCRHI14b, dCRHI14a, dCRHGITL15, dCRHIHTL17b]. Besides, out of the main topic of this thesis but also on the topic of recommendation systems, we have collaborated with work developed by the University of Modena and Reggio Emilia (Italy) [CGI+15, CGI+17].

Concerning other parameters that measure our contributions, one of our papers [dCRHI16], where we focus on the proposed architecture and pull-based context-aware recommendations for mobile environments, is cited in a recent PhD thesis defended at the University of Macedonia (Greece) [Pol17]. As we have also done, the author emphasizes that one of the major issues identified in the literature of context-aware recommendation systems is the lack of good-quality datasets and that most datasets are domain specific. Other works where references to our papers can be found are [YLS+16, PGPS16, MYDÁlG+16, NACH17, PG17, SS17a, SS17b, NWS17, HKLK17, MPCB16, PGPS15, BH16, ATM17, YLY+16], where we can highlight references from journals such as IEEE Communications Surveys and Tutorials [YLS+16], Computer Standards & Interfaces [PG17], the International Journal of Distributed Sensor Networks [HKLK17], and Wireless Personal Communications [NWS17].

9.3 Future Work

The field of context-aware recommendation systems is still a recent area of research. This thesis constitutes a step forward in this area, but there are still interesting research avenues to explore. In the following, we mention some aspects that we would like to consider:

- We would like to continue the development of our architecture, with a special emphasis on the design, implementation and evaluation of the Sentiment Analyzer, User Reliability Analyzer, Rating Reliability Analyzer, and Sensing Engine modules. Besides, we plan to implement the proposed Push-Based Recommendation module and evaluate it in several scenarios. Afterwards, we would like to release the implemented framework as an open source project.
- We could study the use of ontologies to encode knowledge about the different types of items and scenarios as well as rules to define actions to perform with context changes.
- We plan to perform additional tests considering other application scenarios.
- We would like to study and incorporate users' privacy-preserving methods for context-aware recommendation systems in the proposed architecture.

- Regarding DataGenCARS, we plan to develop a graphical user interface to facilitate the definition of the different configuration files and schemas exploited by DataGenCARS and its use. Besides, we would like to incorporate strategies for the generation of dynamic events (e.g., road blocked, traffic jam, parking space available, special offer in the context of shopping recommendations, etc.) that may be taken into account by advanced context-aware recommendation systems during recommendation processes targeted at mobile users. Finally, we could incorporate more advanced data generation strategies, for example to represent effects like the temporality of user profiles (changes of the user preferences along time) or the overall loss of interest in specific items that may go out-of-fashion.

In general, the goal of our future lines of research will be to continue exploring the challenges related to the development and use of context-aware recommendations systems in mobile computing scenarios.

Appendix A

Relevant Classes of the Prototype of the Architecture

In this appendix, we provide more details of the prototype of the proposed architecture, described in Chapter 3. The main modules in our prototype are the following:

- *RepositoryManager* (see Figure A.1). It contains the *DBConnection* class, to establish the connection to a relational database, as well as the *DataAccessLayer* class, that implements the *DataAccess* interface and allows access to the data stored in a database, hiding the details of the procedure needed to access data (use of SQL queries) to other modules.

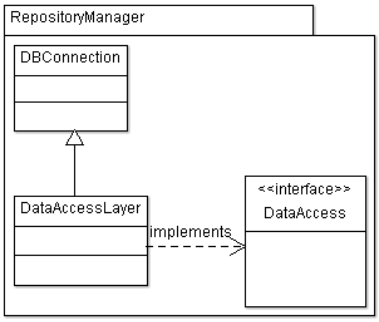


Figure A.1: Class diagram of the *RepositoryManager* module.

- *UserProfileAndContextManager* (see Figure A.2). It contains the *DBDataModel* class, that extends the *AbstractDataModel* class of Apache Mahout, to access not only data of users, items and ratings, but also context data. The resulting data (obtained by using the methods contained in this class) can then be exploited in different recommendation algorithms. This module uses the *Repository Manager* module.

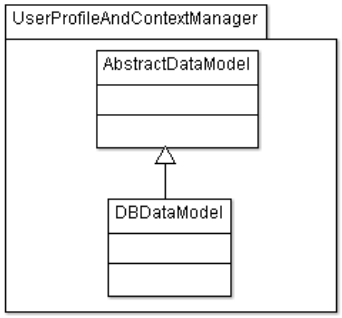


Figure A.2: Class diagram of the *UserProfileAndContextManager* module.

- *Recommendation*. It contains the following modules: *TraditionalRecommendation*, *HybridRecommendation*, *ContextAwareRecommendation*, and *PullBasedRecommendation*. These modules extend the *AbstractRecommender* class and implement the *Recommender* interface of Apache Mahout. In addition, each one of these modules make use of the *UserProfileAndContextManager* module.

- *TraditionalRecommendation* (see Figure A.3). It contains the *PopularityBasedRecommendation* and *ContentBasedRecommendation* classes. The first class recommends the most popular items. The second class applies a traditional content-based recommendation, by using the cosine similarity measure. The design of this module uses and follows the same class structure provided in Apache Mahout [Apa14]. Hence, the collaborative filtering recommendation classes of Apache Mahout can also be used from our prototype. In addition, we include the *CosineItemSimilarity* class in our prototype, which implements the cosine similarity measure. This class extends the *AbstractItemSimilarity* abstract class of Apache Mahout, and can be used to determine the similarity between items, by considering features that describe them.

- *ContextAwareRecommendation* (shown in Figure 8.2, in Chapter 8). It contains the *PrefilteringBasedRecommendation*, *PostfilteringBasedRecommendation*, and *ContextualModelingBasedRecommendation* classes, which implement the different context-aware recommendation paradigms (described in Section 2.3.1) and extend the *AbstractContextAwareRecommendation* abstract class.

- *PullBasedRecommendation* (see Figure A.4). It contains the *PullBasedRecommendation* class, that implements the pull-based recommendation approach (explained in Section 4.1). This class uses the *ContextAwareRecommendation* module.

- *KeywordSearch*. It contains the *HMMBasedSearch* and *IRBasedSearch* classes, and it can be used by the *PullBasedRecommendation* class for the identification of the type of item required by the user, by using keywords (see Section 4.1.6).

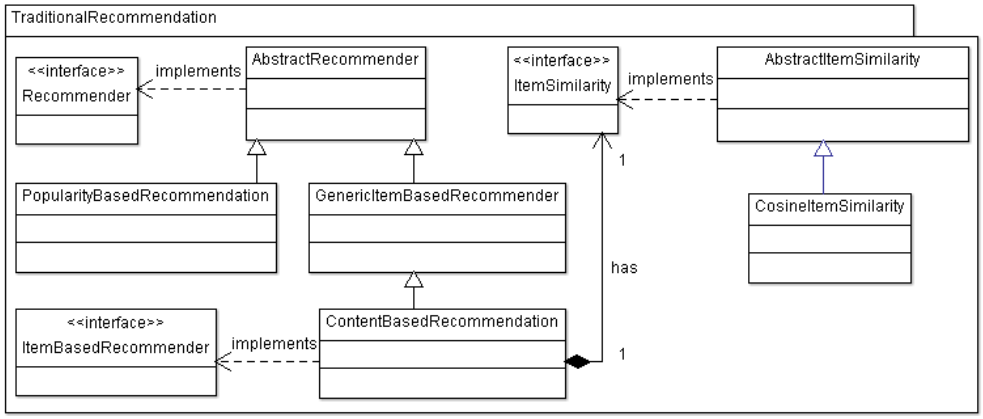


Figure A.3: Class diagram of the *TraditionalRecommendation* module.

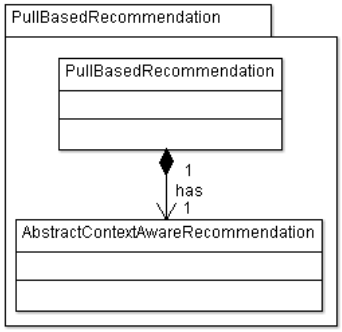


Figure A.4: Class diagram of the *PullBasedRecommendation* module.

- *HybridRecommendation* (see Figure A.5). It contains the *AbstractHybridRecommendation* abstract class, which can be implemented by several types of hybridization (see Section 2.2.3). However, in this prototype we only implemented the weighted hybridization type (in the *AbstractWeightedStrategy* class). In this class, the predicted rating of an item is determined from the results of all the recommendation models (of the *Recommendation* module) applied separately. The *AbstractWeightedStrategy* abstract class is extended by the *WeightedAdditiveStrategy*, *WeightedAverageStrategy*, *WeightedMinimumStrategy*, and *WeightedMaximumStrategy* implementation classes, that determine the sum, average, minimum, or maximum, respectively, of all the item ratings estimated by the recommendation models applied.

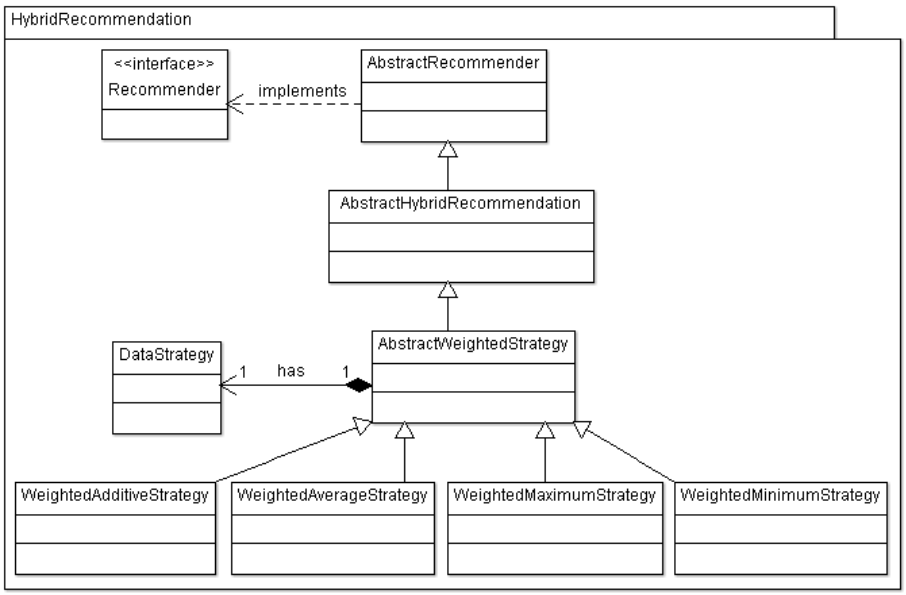


Figure A.5: Class diagram of the *HybridRecommendation* module.

Appendix B

DataGenCARS: Architecture Details

In this appendix, we present details of the DataGenCARS architecture, described in Chapter 5. In Section B.1, we explain the definition, structure and contents of input files (e.g., schema, profile and configuration files) required by DataGenCARS for the generation of synthetic data. In Section B.2, we present the output files (e.g., user, item, context and rating files) generated by DataGenCARS. In Section B.3, we show a detailed view of the architecture and the classes that compose it.

B.1 Input Files for DataGenCARS

The input files are text files with a human-readable format. In the following, we summarize the structure and contents of the input files used to define schemas (user schemas, types of items, and contexts) and dataset generation parameters.

B.1.1 Definition of Schemas

User schemas, contexts, and types of items, are defined by means of three text files: *user_schema* (see Figure B.1), *context_schema* (see Figure B.3), and *item_schema* (see Figure B.2, where for simplicity a single attribute is considered), respectively. Each schema is composed by a list of attributes (attribute names and types/domains) and references to Java class generators that will generate values for the different attributes. The user schema also includes an attribute representing the user profile (each generated user will be assigned one among the different user profiles defined). The schema of the types of items also includes an attribute for the item profile (needed to ensure the consistency among attribute values, as explained in B.1.2), as well as additional constraints such as the specification of *uniqueness* for the generated attribute values (if required).

```
type=user
number_attributes=4

name_attribute_1=age
type_attribute_1=Integer
minimum_value_attribute_1=15
maximum_value_attribute_1=63
generator_type_attribute_1=data.generator.attribute.RandomAttributeGenerator

name_attribute_2=sex
type_attribute_2=String
number_posible_values_attribute_2=2
posible_value_1_attribute_2=F
posible_value_2_attribute_2=M
generator_type_attribute_2=data.generator.attribute.RandomAttributeGenerator

name_attribute_3=city
type_attribute_3=String
number_posible_values_attribute_3=3
posible_value_1_attribute_3=Zaragoza
posible_value_2_attribute_3=Granada
posible_value_3_attribute_3=Valencia
generator_type_attribute_3=data.generator.attribute.RandomAttributeGenerator

name_attribute_4=country
type_attribute_4=String
generator_type_attribute_4=data.generator.attribute.FixedAttributeGenerator
input_parameter_attribute_4=United States # all the users live in the USA
```

Figure B.1: Simplified example of a user schema file.

B.1.2 Definition of Profiles

user_profile: is a comma-separated values (CSV) file that describes the different user profiles desired (see Figure B.4). Each line contains the description of a user profile (i.e., the utility function of the user), which consists of the identifier of the user profile and a sequence of weights to apply to the values of each of the relevant attributes defining the context of the user and the type of item. To better emphasize the relative importance of each attribute, the sum of all the weights should add up to one. DataGenCARS includes a functionality for automatic readjustment or normalization of the weights, that respects the desired relative importance of each attribute.

The domain of values of each (numeric or categorical) attribute is assumed to be ordered in a meaningful way, but some users may prefer lower values for a given attribute and others may prefer higher values. Therefore, next to each weight, it is

```
type=item
number_attributes=1

name_attribute_1=director_movieCountry
type_attribute_1=AttributeComposite
number_maximum_subattribute_attribute_1=2
name_subattribute_1_attribute_1=director
name_subattribute_2_attribute_1=movieCountry
generator_type_attribute_1=data.generator.attribute.RandomLineAttributeGenerator
input_parameter_attribute_1=movie_information.csv
important_weight_attribute_1=true
```

Figure B.2: Simplified example of an item schema file.

possible to identify if higher (+) or lower (−) values of the corresponding attribute are preferred. This provides significant flexibility, as the only requirement is the need to define the range of values of each relevant attribute in any significant order: some user profiles may prefer lower values of the range (−) and other profiles could rather choose higher values (+). For instance, for the attribute "ambiance" of a restaurant and a range of values defined as "extremely calm", "calm", "normal", "lively", "very lively", some users may prefer calm environments while others may have a tendency towards trendy and active environments.

It should be noted that, in general, not all the attributes of the context and types of item are *relevant* for the user profile (e.g., the specific name of an item of type "hotel", for example, is usually irrelevant). Consequently, the schemas of items and contexts (item_schema and context_schema) include labels to mark the attributes that are relevant to define the user profile. Moreover, the header of the file user_profile indicates the names of the relevant attributes that will be considered in the profiles defined in the following lines of the file.

item_profile: is a text file that describes the different profiles defined for the generation of values of attributes in a consistent way (see Figure B.5). As an example, this file allows the definition of a profile "good" for a restaurant indicating that in such a profile we would expect, in general, values "excellent" or "good" for the attributes "quality of food" and "quality of service". DataGenCARS enables a flexible definition of the different profiles required, for example allowing overlapping among values (e.g., "good" may be an expected/suitable value of an attribute for both the profiles "good" and "normal").

B.1.3 Configuration of a Dataset Generation Process

generation_config: is a text file that contains parameters to configure the dataset generation process (see Figure B.6). For example, it allows defining: the percentage of items to generate for each item profile; the percentage of users to generate for

```
type=context
number_attributes=3

name_attribute_1=time
type_attribute_1=String
number_possible_values_attribute_1=4
possible_value_1_attribute_1=morning
possible_value_2_attribute_1=afternoon
possible_value_3_attribute_1=evening
possible_value_4_attribute_1=night
generator_type_attribute_1=data.generator.attribute.RandomAttributeGenerator
important_weight_attribute_1=true

name_attribute_2=season
type_attribute_2=String
number_possible_values_attribute_2=4
possible_value_1_attribute_2=spring
possible_value_2_attribute_2=summer
possible_value_3_attribute_2=autumn
possible_value_4_attribute_2=winter
generator_type_attribute_2=data.generator.attribute.RandomAttributeGenerator
important_weight_attribute_2=true

name_attribute_3=companion
type_attribute_3=ArrayList
number_maximum_component_attribute_3=4
type_component_attribute_3=Boolean
component_1_attribute_3=friends
component_2_attribute_3=family
component_3_attribute_3=girlfriend
component_4_attribute_3=children
generator_type_attribute_3=data.generator.attribute.BooleanArrayListAttributeGenerator
input_parameter_attribute_3=2 # at least two of the components will be false
```

Figure B.3: Simplified example of a context schema file.

```
userProfileID; director; movieCountry; time;     season;   other
1;             (-) 0.4;  (-) 0.4;      0;        0;        0.2
2;             (-) 0.2;  0;            (-) 0.2;  (+) 0.6;  0
3;             0;        (-) 0.6;      (-) 0.4;  0;        0
...
```

Figure B.4: Simplified example of a user profile file.

```
number_profiles=3

name_profile_1=good
name_profile_2=normal
name_profile_3=bad

ranking_order_profile=desc
overlap_midpoint_left_profile=1
overlap_midpoint_right_profile=1
```

Figure B.5: Simplified example of an item profile file.

```
number_user=100
number_item=250
number_context=40
number_rating=500

probability_percentage_profile_1=10
probability_percentage_profile_2=30
probability_percentage_profile_3=60

noise_percentage_profile_1=20
noise_percentage_profile_2=20
noise_percentage_profile_3=20

percentage_rating_variation=25
```

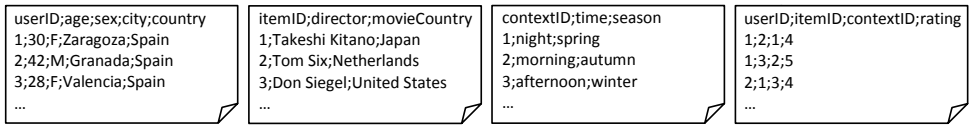
Figure B.6: Simplified example of a generation configuration file.

each user profile; the number of users, items, contexts, and ratings desired; the noise to introduce in the definition of user profiles; or the impact of user expectations in future ratings (maximum percentage of variation in the expected rating and number of recent ratings in the previous history that may have an impact).

B.2 Output Files for DataGenCARS

As shown in Figure B.1, DataGenCARS generates the following CSV files: *user* (U), *item* (I), *context* (C), and *ratings* (U x I x C $\rightarrow$ ratings). The first line of each file is a header (with the names of the fields defining the file) and each of the remaining lines represents the information of a generated user, item, context, and rating, respectively. The output files for users, items and contexts have a structure defined by the corre-

sponding schema (user schema, item schema, and context schema, respectively). The output file for ratings contains a user identifier, a context identifier, an item identifier, a rating, and the date and time when the user provided the rating.



(a) user.csv (b) item.csv (c) context.csv (d) ratings.csv

Figure B.1: Simplified examples of output files.

B.3 Relevant Classes in the Architecture

The main classes of the architecture were described in Section 5.1.2 and Figure B.1 shows a detailed view of the architecture. In the following, we briefly comment on some other interesting classes:

- Some classes in Figure B.1 are focused on the extraction of geographic information from OpenStreetMap. For example, *PlacesFromPOI* obtains items of a specific type that are close to a specific geographic location (instances of the Java class *Place*). *HTTPRequestPoster* uses *Nominatim* [Ope04] to obtain structured data about specific items. *XMLQueryProcessing* is an auxiliary class (based on the Xerces Java Parser) that parses XML data. Overall, these classes support the obtention of real data that can facilitate the creation of more realistic synthetic datasets (or mixed real-synthetic datasets). For example, the information obtained from OpenStreetMap can be stored in a file to be used later for the dataset generation.

- Other classes provide support for automatic conversion from Java classes to item schema files (class *GenerateSchemaFromClass*) and vice versa (class *GenerateClassFromSchema*).

- *AttributeComposite* is used to represent sets of attributes whose values must be generated jointly for consistency reasons (as explained in Section 5.1.1).

- Finally, other interesting classes include *GenerateSampleDataset*, *GenerateUserProfile*, *ReplaceUnknownValues*, and *SplitDatasetInSeveralFiles*, whose purposes were made clear in Section 5.2.

We would like to highlight the generality of the proposed architecture, that enables the generation of different types of items and scenarios by appropriately defining the input data files. Thus, for example, it is possible to define any schema for items, users, and contexts. The data generation strategies can also be easily configured and adapted

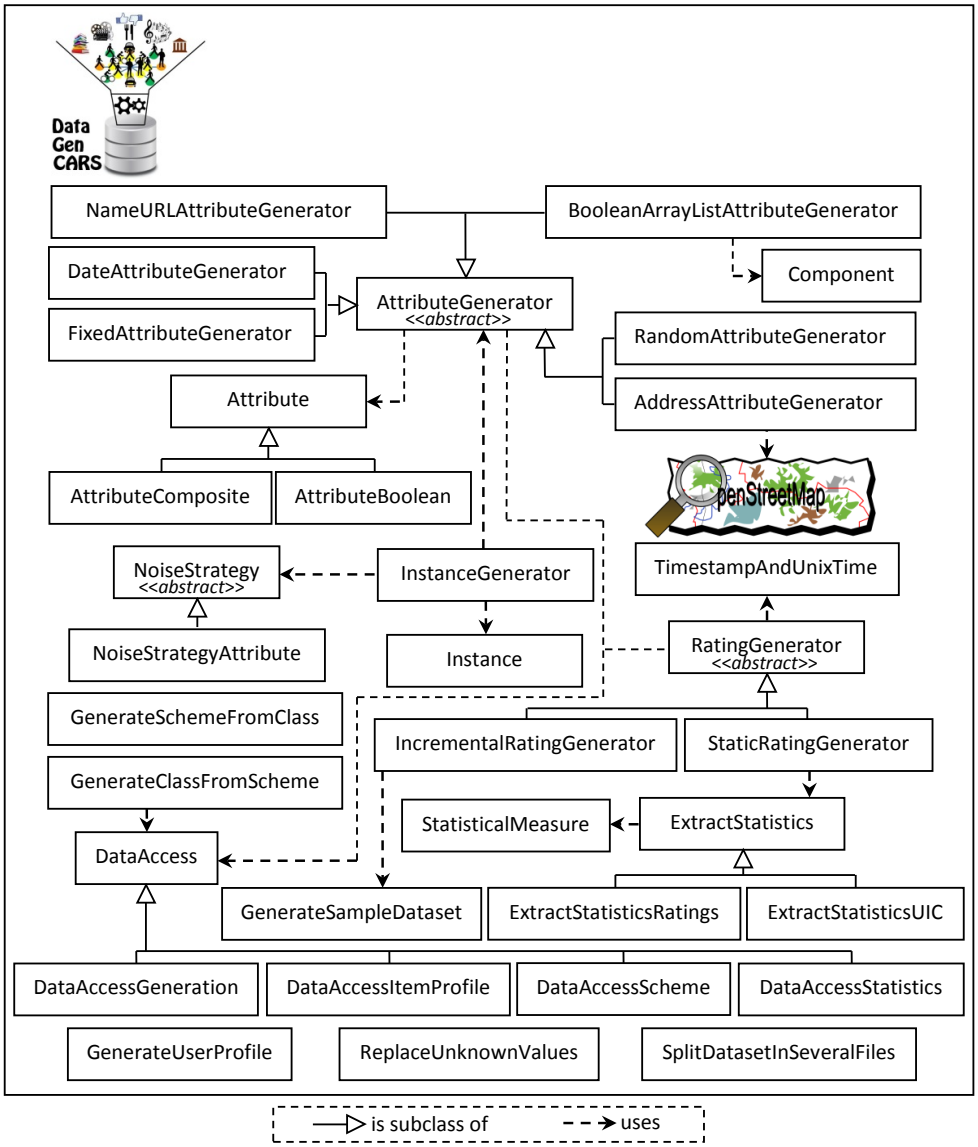


Figure B.1: Detailed high-level class diagram.

by extending the appropriate classes and referencing them in the corresponding input files. The use of the Java Reflection API plays a key role, as it allows creating instances of classes and invoking methods dynamically at runtime without the need to predefine the specific needs at compilation-time.

Appendix C

Evaluation of Keyword-Based Item Type Searching Approaches

In this section, we present the experimental evaluation of the HMM-based and IR-based methods (see Section 4.1.6), implemented in the prototype of our architecture for the identification of the type of item required by a user in a pull-based recommendation process. The user types some keywords and the system needs to infer the type of item required. In Section C.1, we describe the datasets that we use for the evaluation. Besides, we present the keyword-based queries that are evaluated. In Section C.2, we discuss the accuracy obtained with both proposals. In Section C.3, we analyze the impact of increasing the number of instances in the datasets on the performance of the methods proposed. In Section C.4, we present an evaluation of the use of computational resources (in terms of the time required) to identify the item type requested by the user. Finally, in Section C.5, we analyze the impact of the model parameters used on the HMM approach.

C.1 Datasets and Keyword-Based Queries

We performed an experimental evaluation to compare the HMM-based and IR-based approaches proposed for the identification of the type of item required by a user in a pull-based recommendation process. For this, we consider the following datasets: LDOS-CoMoDa [KOK+11], InCarMusic [BKL+11], Book-crossing [ZMKL05], ConcertTweets [AT14], RCdata [VGGSPM11] and Frappe [BCKO15]. Specifically, we focus only on information related to items. In Table C.1, some of the statistics related to the items of these datasets are described.

In order to represent the HMM model, we used the item types, the feature names, and the features values of the six datasets considered. However, we only chose the

	LDOS-CoMoDa	InCarMusic	Book-crossing	ConcertTweets	RCdata	Frappe
Number of items	2513	139	271,084	50,971	130	4082
Number of attributes of the items	8	7	7	8	25	10

Table C.1: Dataset statistics.

most appropriate features (see Table C.2). Specifically, we ignore some features (name and values) of the following datasets: InCarMusic (e.g., album, mp3url, description and imageurl), Book-crossing (e.g., image-URL-S, image-URL-M, and image-URL-L), ConcertTweets (e.g., URL), RCdata (e.g., fax, URL, and the-geom-meter), and Frappe (e.g., icon, description, and short description). We decided to ignore these features because they do not provide useful information (e.g., there are attributes where all their values are unknown, some URLs are incomplete, etc.). We use the information of 1000 instances, selected randomly, for each of those datasets.

Considering the six datasets mentioned, the HMM model λ is composed by 52 states (e.g., film_director, music_artist, book_title, concert_date, restaurant_address, application_category). These states are the combination of the item types (e.g., film, music, book, concert, restaurant and application) and the feature names (e.g., director, artist, title, date, address, category, etc.) of the six datasets.

The two methods proposed were evaluated by using 45 queries (see Table C.3). The generated queries are a representative set of the item information contained in the six databases. Notice that some queries actually correspond to item types that are not available in the datasets considered (so the best possible output is "other", that represents a type of item not identified). For example, queries with identifiers from 30 to 35 explicitly include elements that are not part of the contents of the dataset.

C.2 Accuracy

The first proposed solution based on the HMM computes the most likely state sequence matching an observation sequence given an HMM model. The second proposal (based on IR) searches the keywords in the query in the index of documents and returns a ranked list of hits.

The results of the queries defined in Table C.3 are displayed in Table C.4 by using both proposals. According to the values obtained of precision, recall and F1 measure in Table C.5 and Table C.6, the HMM model performs better than the IR model in the experimental setup considered. In Table C.6, the accuracy for *other* is zero because the IR model never succeeds with this type of item (see Table C.4).

C.3 Impact of the Number of the Instances

We conducted another experiment with the aim of analyzing the impact of increasing the number of instances in the datasets on the performance of the two methods analyzed. For this experiment, we specifically selected the datasets Book-crossing,

Dataset	Item type	Feature name	Number of different values for each attribute
LDOS-CoMoDa	film	director	814
		country	39
		language	27
		year	63
		genre	23
		actor	1997
		budget	174
		title	1581
InCarMusic	music	artist	119
		category	10
		title	138
Book-crossing	book	isbn	271,084
		title	238,982
		author	99,361
		year	116
		publisher	16,596
ConcertTweets	concert	date	673
		city	4741
		state	161
		latitude	17,219
		longitude	26,438
		venue	22,759
		band	13,605
RCdata	restaurant	accessibility	3
		address	99
		alcohol	3
		ambience	2
		area	2
		city	6
		country	1
		cuisine	59
		days	7
		dress	3
		franchise	2
		hours	274
		latitude	129
		longitude	129
		name	126
		parking	7
		payment	11
		price	3
		services	3
		smoking	5
		state	4
		zip	34
Frappe	application	category	31
		developer	2806
		downloads	16
		language	28
		name	3503
		package	3531
		rating	225

Table C.2: Information about the datasets considered for the evaluation of keyword-based approaches.

Query Id	Original query
1	films similar to “toy story”
2	a romantic movie of the year 2009
3	videos of the director “walter lang”
4	the english film titled monkeys
5	a movie of the actor “diane ladd”
6	a music of the singer giovanni
7	rock song
8	music titled “für immer”
9	song of a rock artist
10	songs like “potato head blues” by “louis armstrong”
11	books about “seabiscuit”
12	books similar to ”fast women” by the author “jennifer crusie”
13	publications with an isbn number similar to 195153448
14	documents by the publisher scholastic
15	books with title “urban etiquette” and publisher ”wildcat canyon
16	concert of the band “iron maiden”
17	musical group that will play on ”02/04/2014” in Madrid
18	concerts in the venue “twickenham stadium”
19	the band direction in the state germany
20	concerts like “cattle decapitation” in ”cellular center”
21	publications with an isbn
22	place to eat
23	lodging in Modena
24	romantic melody
25	upcoming soccer matches in Barcelona
26	a recent horror movie
27	songs of movies
28	readings about movies
29	books about singers
30	self-help documents
31	romantic movie in 1949
32	policy documents of 1930
33	festivals in the region of the Holguin
34	movies that were premiered in 1927
35	documents of the Antarctica of 1908
36	restaurant with bar and permit smoking
37	place for dinner with an ambience familiar and low price
38	places opened in the hours of “12-00-22-00” to have lunch
39	restaurants with the name “taqueria el amigo”
40	french food and with “MasterCard Eurocard” payment
41	applications of photography
42	mobile applications developed by yahoo
43	chats similars to “whatsapp messenger”
44	“sport game” with many downloads
45	a apk similar to “Angry Birds” and language es

Table C.3: Information about the queries considered for the evaluation of keyword-based approaches.

Query Id	Real item type	Item type by HMM	Item type by IR
1-5 and 26	film	film	film
6	music	music	film
7, 9, and 10	music	music	music
8	music	music	film
11, 12, 14, 15, and 29	book	book	book
13	book	book	restaurant
16, 18, 19, and 20	concert	concert	concert
17	concert	concert	application
21	book	book	restaurant
22	other	other	concert
23	other	other	book
24	music	other	book
25	other	concert	application
27	music	music	film
28	book	concert	book
30	book	other	other
31	film	other	application
32	book	other	book
33	concert	other	concert
34	film	other	application
35	book	other	other
36-40	restaurant	restaurant	restaurant
41-45	application	application	application

Table C.4: Results to the queries with both keyword-based models.

Item type	Precision	Recall	F1
film	1.0	0.75	0.86
music	1.0	0.86	0.92
book	1.0	0.64	0.78
concert	0.71	0.83	0.77
restaurant	0.83	1.0	0.91
application	1.0	1.0	1.0
other	0.25	0.67	0.36
Average	0.83	0.82	0.80

Table C.5: Evaluation of the HMM model.

Item type	Precision	Recall	F1
film	0.67	0.75	0.71
music	1.0	0.43	0.6
book	0.78	0.64	0.7
concert	0.83	0.83	0.83
restaurant	0.71	1.0	0.71
application	0.56	1.0	0.86
other	0.0	0.0	0.0
Average	0.65	0.66	0.63

Table C.6: Evaluation of the IR model.

ConcertTweets, and Frappe, which contain the larger number of instances (see Table C.1). For each dataset, we considered four different versions with an increasing number of instances (1000, 2000, 3000, and 4000 instances).

Then, we evaluated the performance (precision, recall and F1-measure) of both models, as shown in Figures C.1, C.2, and C.3. As shown in the figures, increasing the number of instances in the datasets, in general, leads to a decrease in the performance of both methods, but the worsening is quite moderate and not very significant beyond 2000 instances per dataset. Again, in general, the HMM-based method performs better than the IR-based approach.

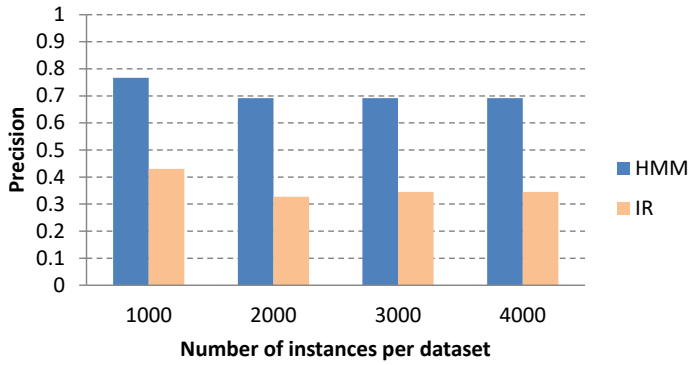


Figure C.1: Average precision of both keyword-based models for several numbers of instances.

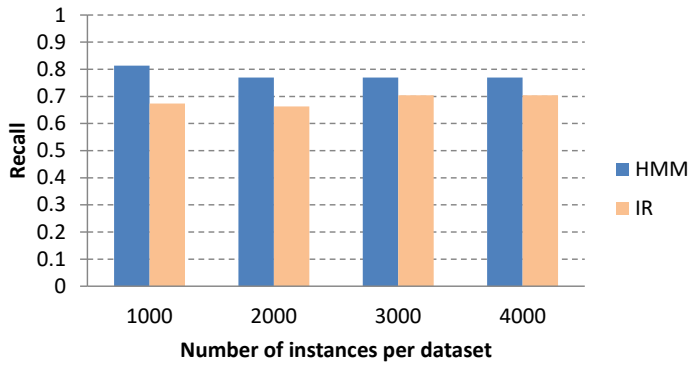


Figure C.2: Average recall of both keyword-based models for several numbers of instances.

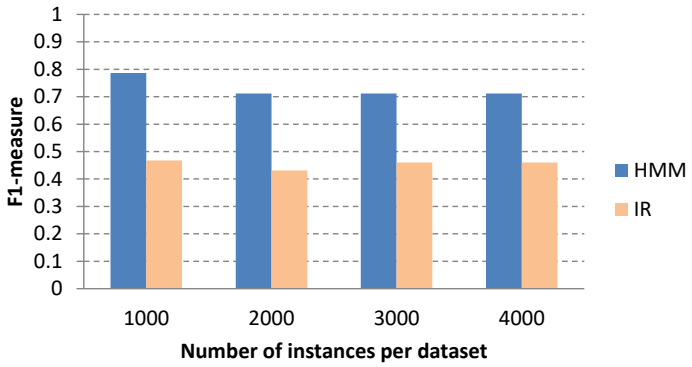


Figure C.3: Average F1-measure of both keyword-based models for several numbers of instances.

C.4 Use of Computational Resources

For each query, we also performed an evaluation of the processing time required. The evaluation results (see Figures C.4 and C.5) show that the HMM model is more efficient than the IR model, it requires less time to identify the item type required by the user. Nevertheless, in both cases we have processing times of the order of just a few milliseconds for the datasets considered.

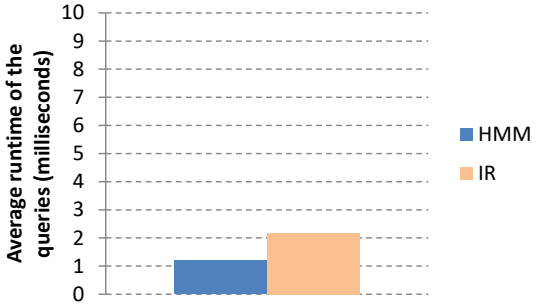


Figure C.4: Average runtime of the queries for both keyword-based models.

In Table C.7, we show the average runtime of the queries and the average time needed for the creation of both models. The internal implementation of the functionalities for HMM and IR provided by Mahout [Apa14] and Lucene [Apa05] seems to include some initialization overhead that affects only the first query submitted, as reflected in Table C.7; these first-query initialization times are not accumulated in the results shown in Figures C.4 and C.5, as we rather consider them as initial overheads.

However, HMM uses a 39.38% of additional memory (in megabytes, MB) that the IR model (see Table C.8).

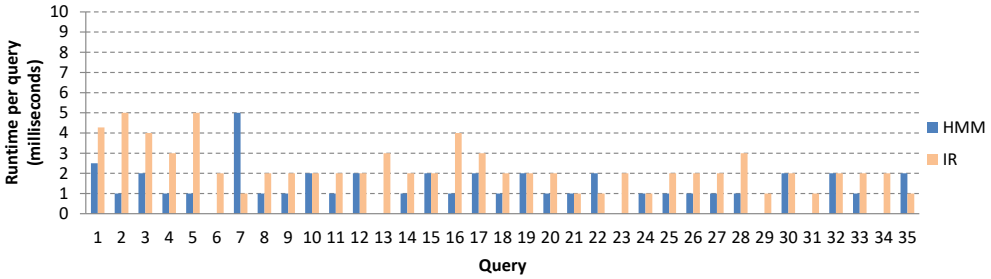


Figure C.5: Runtime per query for both keyword-based models.

	HMM	IR
Average runtime per query	1.64 ms	3.91 ms
Average time for model creation	0.11 ms	1.53 ms
Average initialization time for the first query	2.0 ms	36.0 ms

Table C.7: Runtime per query and the process to create the underlying model.

	Memory usage with HMM	Memory usage with IR
Average	9.54 MB	2.86 MB
Maximum	11.02 MB	4.92 MB
Minimum	7.31 MB	0.91 MB
Average	61.55 %	18.44 %
Maximum	71.15 %	31.77 %
Minimum	47.22 %	5.85 %

Table C.8: Memory usage of the models.

C.5 Impact of the Model Parameters Used in HMM

The experimental results obtained are quite promising. A potential problem with the HMM model is how to determine suitable probability values when the observation vector size is very large; potentially, it could happen that very small probabilities could be rounded to 0 if the global probability values are shared among many different possible values. Besides, other methods to assign the probabilities (by default we consider a proportional distribution/sharing) could be applied. Despite these concerns, the performance results obtained with the datasets evaluated so far are quite good.

Nevertheless, we consider interesting to evaluate the impact of modifying the parameters of HMM. In this case, we modify the observation probabilities B manually. Specifically, we duplicate the probability values of the top k (e.g., k=2) most relevant observations (such as the item type and the attribute representing the name of the

item) for each state. The system is responsible for automatically adjusting the different values ensuring that the sum of all the probabilities is one. In this case, the idea of this strategy is to model observations whose likelihood depends on the state.

In Figure C.6, we show the average performance of two HMMs (i.e., one without the parameters adjusted and the other one with the parameters modified and adjusted automatically), by using the queries presented in Table C.9. These specific queries contain information about the parameters that are expected to affect the model (the item type and attribute name of the item), so we can easily see the impact of these parameters on the model. According to the results shown in Figure C.6, the HMM-based method performs better when the parameters are appropriately adjusted.

Query Id	Original query
1	book 2002
2	film 2002
3	concert mexico
4	rock category
5	rock category
6	"sport game" category
7	film "diario ana frank"
8	book "ann frank remembered"
9	book "ana frank"
10	film "ann frank"

Table C.9: Information about the queries for the experiment where the HMM parameters are modified.

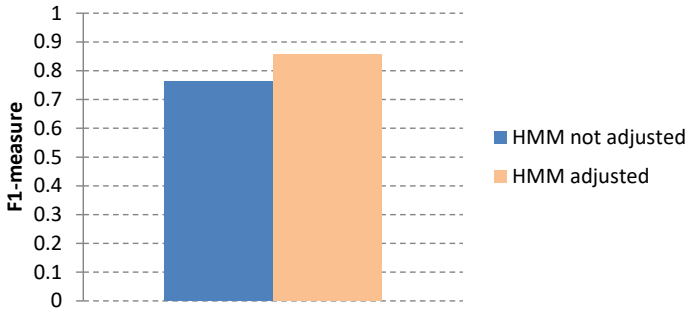


Figure C.6: Average performance of the HMM keyword-based model for the default and adjusted parameters.

Relevant Publications Related to this Thesis

Journals

[dCRHI16] M. del Carmen Rodríguez-Hernández and S. Ilarri. Pull-based recommendations in mobile environments. *Computer Standards & Interfaces*, 44:185–204, February 2016. ISSN 0920-5489. DOI 10.1016/j.csi.2015.08.002.

[dCRHIHTL17a] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. DataGenCARS: A Generator of Synthetic Data for the Evaluation of Context-Aware Recommendation Systems. *Pervasive and Mobile Computing*, 38, Part 2(2017):516–541, July 2017. ISSN 1574-1192. DOI 10.1016/j.pmcj.2016.09.020. Special Issue on Context-aware Mobile Recommender Systems.

[IHTLdCRH15] S. Ilarri, R. Hermoso, R. Trillo-Lado, and M. del Carmen Rodríguez-Hernández. A review of the role of sensors in mobile context-aware recommendation systems. *International Journal of Distributed Sensor Networks*, 11(11):1–30, January 2015. ISSN 1550-1329. DOI 10.1155/2015/489264.

International Conferences

[dCRHI14c] M. del Carmen Rodríguez-Hernández and S. Ilarri. Towards a context-aware mobile recommendation architecture. In I. Awan, M. Younas, X. Franch, and C. Quer, editors, *11th International Conference on Mobile Web Information Systems (MobiWIS)*, volume 8640 of *Lecture Notes in Computer Science (LNCS)*, pp. 56–70. Springer, Cham, August 2014. ISBN 978-3-319-10359-4. DOI 10.1007/978-3-319-10359-4_5.

[HITdCRH15] R. Hermoso, S. Ilarri, R. Trillo, and M. del Carmen Rodríguez-Hernández. Push-Based Recommendations in Mobile Computing Using a Multi-Layer Contextual Approach. In *13th International Conference on Advances in Mobile Computing and Multimedia (MoMM)*, pp. 149–158. ACM, New York (USA), December 2015. ISBN 978-1-4503-3493-8. DOI 10.1145/2837126.2837128.

[dCRHGITL16] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo-Lado, and F. Guerra. Towards keyword-based pull recommendation systems. In *18th International Conference on Enterprise Information Systems (ICEIS)*, volume 1, pp. 207–214. SCITEPRESS (Science and Technology Publications, Lda.), April 2016. ISBN 978-989-758-187-8. DOI 10.5220/0005865402070214.

[dCRHIHTL17c] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. Towards trajectory-based recommendations in museums: Evaluation of strategies using mixed synthetic and real data. In *Eighth International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN)*, volume 113, pp. 234–239. September 2017. ISSN 1877-0509. DOI 10.1016/j.procs.2017.08.355. Procedia Computer Science.

[dCRHITH17] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo, and R. Hermoso. Context-aware recommendations using mobile P2P. In *15th International Conference on Advances in Mobile Computing and Multimedia (MoMM)*. ACM, New York (USA), December 2017. ISBN 978-1-4503-5300-7. DOI 10.1145/3151848.3151856. 10 pp., accepted, to appear.

International Workshops

[dCRHITLH15] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo-Lado, and R. Hermoso. Location-aware recommendation systems: Where we are and where we recommend to go. In *International Workshop on Location-Aware Recommendations (LocalRec), in conjunction with the Ninth ACM Conference on Recommender Systems (RecSys)*, volume 1405, pp. 1–8. CEUR Workshop Proceedings, September 2015. ISSN 1613-0073.

National Conferences

[dCRHI14a] M. del Carmen Rodríguez-Hernández and S. Ilarri. Context-Aware Recommendations in Mobile Environments. In J. Tuya, M. Ruiz,

and N. Hurtado, editors, *XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pp. 85–90. September 2014. ISBN 978-84-697-1152-1, 84-697-1152-0.

[dCRHGITL15] M. del Carmen Rodríguez-Hernández, F. Guerra, S. Ilarri, and R. Trillo-Lado. A first step towards keyword-based searching for recommendation systems. In *XX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*. September 2015. 4 pp., `http://hdl.handle.net/11705/JISBD/2015/007`.

[dCRHIHTL17b] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. Definiendo un caso de estudio para recomendaciones dinámicas móviles. In *XXII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*. July 2017. 4 pp., `http://hdl.handle.net/11705/JISBD/2017/013`.

Local Events

[dCRHI14b] M. del Carmen Rodríguez-Hernández and S. Ilarri. Researching Context-Aware Recommendation Systems in Mobile Environments. *Actas de las III Jornadas de Jóvenes Investigadores del I3A*, 2:59–60, June 2014. ISSN 2341-4790.

Other Collaborations Related to Recommendation Systems

[CGI+17] S. Cadegnani, F. Guerra, S. Ilarri, M. del Carmen Rodríguez-Hernández, R. Trillo-Lado, et al. Exploiting linguistic analysis on URLs for recommending web pages: a comparative study. In N. T. Nguyen, R. Kowalczyk, A. M. Pinto, and J. Cardoso, editors, *Transactions on Computational Collective Intelligence XXVI*, volume 10190 of *Lecture Notes in Computer Science (LNCS)*, pp. 26–45. Springer, Cham, 2017. ISBN 978-3-319-59268-8. DOI 10.1007/978-3-319-59268-8_2.

[CGI+15] S. Cadegnani, F. Guerra, S. Ilarri, M. del Carmen Rodríguez-Hernández, R. Trillo-Lado, et al. Recommending web pages using item-based collaborative filtering approaches. In J. Cardoso, F. Guerra, G.-J. Houben, A. M. Pinto, and Y. Velegrakis, editors, *Semantic Keyword-based Search on Structured Data Sources: First COST Action IC1302 International KEYSTONE Conference (IKC)*, volume 9398 of *Lecture Notes in Computer Science*

(LNCS), pp. 17–29. Springer, September 2015. ISBN 978-3-319-27932-9. DOI 10.1007/978-3-319-27932-9_2.

References

[AA14] A. Agarwal and K. Agarwal. The next generation mobile wireless cellular networks–4G and beyond. *American Journal of Electrical and Electronic Engineering*, 2(3):92–97, April 2014.

[AA16] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *10th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–8. IEEE, January 2016. ISBN 978-1-4673-7807-9.

[AAAPS09] N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. Tell me who I am: An interactive recommendation system. *Theory of Computing systems*, 45(2):261–279, August 2009. ISSN 1433-0490.

[AAHC17] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng. Fog computing for the Internet of Things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, March 2017. ISSN 1089-7801.

[AAP+14] I. Ayala, M. Amor, M. Pinto, L. Fuentes, and N. Gámez. iMuseumA: An agent-based context-aware intelligent museum system. *Sensors*, 14(11):21213–21246, 2014. ISSN 1424-8220.

[ABC+02] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, et al. Banks: Browsing and keyword searching in relational databases. In *28th International Conference on Very Large Data Bases (VLDB)*, pp. 1083–1086. VLDB Endowment, August 2002.

[ABC+05] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, et al. Towards highly adaptive services for mobile computing. In E. Lawrence, B. Pernici, and J. Krogstie, editors, *IFIP TC8 Working Conference on Mobile Information Systems (MOBIS): Mobile Information Systems*, volume 158 of *IFIP International Federation for Information Processing (IFIPAICT)*, pp. 121–134. Springer, Boston, MA, September 2005. ISBN 978-0-387-22874-7.

[ABF16] I. Akermi, M. Boughanem, and R. Faiz. Just-In-Time recommendation approach within a mobile context. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 636–639. IEEE, October 2016. ISBN 978-1-5090-4470-2.

[ABFO08] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. M. Onana. ALAMBIC: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5):307–334, October 2008. ISSN 1615-5270.

[ABK+07] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, et al. DBpedia: A nucleus for a Web of Open Data. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, et al., editors, *The Semantic Web: Sixth International Semantic Web Conference (ISWC) and Second Asian Semantic Web Conference (ASWC)*, volume 4825 of *Lecture Notes in Computer Science (LNCS)*, pp. 722–735. Springer, Berlin, Heidelberg, November 2007. ISBN 978-3-540-76298-0.

[AC10] S. Arlot and A. Celisse. A survey of cross–validation procedures for model selection. *Statistics Surveys*, 4:40–79, 2010.

[ACD02] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *18th International Conference on Data Engineering (ICDE)*, pp. 5–16. IEEE, March 2002. ISBN 0-7695-1531-2. ISSN 1063-6382.

[AD72] J. H. Ahrens and U. Dieter. Computer methods for sampling from the exponential and normal distributions. *Communications of the ACM*, 15(10):873–882, October 1972. ISSN 0001-0782.

[AD15] S. S. Avhad and S. R. Durugkar. LARS∗: Location-aware recommendation system. *International Journal Of Engineering, Education And Technology*, 3(2):1–6, April 2015. ISSN 2320-883X.

[Ada14] P. Adamopoulos. ConcertTweets: A multi-dimensional data set for recommender systems research, 2014. `http://people.stern.nyu.edu/padamopo/data/concertTweets.html`, [Accessed on October 3, 2017].

[ADB+99] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, et al. Towards a better understanding of context and context-awareness. In H.-W. Gellersen, editor, *First International Symposium on Handheld and Ubiquitous Computing (HUC)*, volume 1707 of *Lecture Notes in Computer Science (LNCS)*, pp. 304–307. Springer, Berlin, Heidelberg, September 1999. ISBN 978-3-540-48157-7.

[ADNZ15] E. Ashley-Dejo, S. Ngwira, and T. Zuva. A survey of context-aware recommender system and services. In *International Conference on Computing, Communication and Security (ICCCS)*, pp. 1–6. IEEE, December 2015. ISBN 978-1-4673-9354-6.

[Agg16] C. C. Aggarwal. *Model-Based Collaborative Filtering*, chapter 3, pp. 71–138. Springer, Cham, 2016. ISBN 978-3-319-29659-3.

[AGHI09] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Second ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 5–14. ACM, New York (USA), February 2009. ISBN 978-1-60558-390-7.

[AHD14] C. Anton-Haro and M. Dohler. *Machine-to-machine (M2M) Communications: Architecture, Performance and Applications*. Elsevier, 2014. ISBN 978-1-78242-102-3.

[Ahn08] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008. ISSN 0020-0255.

[AIM10] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, October 2010. ISSN 1389-1286.

[AJ14] G. Adomavicius and D. Jannach. Preface to the special issue on context-aware recommender systems. *User Modeling and User-Adapted Interaction*, 24(1-2):1–5, 2014.

[AJJR14] A. Abbasi, A. Javari, M. Jalili, and H. R. Rabiee. Enhancing precision of Markov-based recommenders using location information. In *Third International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 188–193. IEEE, September 2014. ISBN 978-1-4799-3080-7.

[AjKH06] H. Ahn, K. jae Kim, and I. Han. Mobile advertisement recommender system using collaborative filtering: MAR-CF. In *2006 Conference of the Korea Society of Management Information Systems (KGSF)*, pp. 709–715. 2006.

[AK12] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2012.

[AK15] G. Adomavicius and Y. Kwon. Multi-criteria recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, chapter 24, pp. 847–880. Springer, Boston, MA, 2015. ISBN 978-1-4899-7637-6.

[ALLR16] A. Alti, A. Lakehal, S. Laborie, and P. Roose. Autonomic semantic-based context-aware platform for mobile applications in pervasive environments. *Future Internet*, 8(4):1–48, September 2016. ISSN 1999-5903.

[AM06] S. S. Anand and B. Mobasher. Contextual recommendation. In B. Berendt, A. Hotho, D. Mladenic, and G. Semeraro, editors, *Workshop on Web Mining (WebMine). From Web to Social Web: Discovering and Deploying User and Content Profiles*, volume 4737 of *Lecture Notes in Computer Science (LNCS)*, pp. 142–160. Springer, Berlin, Heidelberg, September 2006. ISBN 978-3-540-74951-6.

[Ama13] X. Amatriain. Big & personal: Data and models behind Netflix recommendations. In *Second International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine)*, pp. 1–6. ACM, New York (USA), August 2013. ISBN 978-1-4503-2324-6.

[AMO13] M.-D. Albakour, C. Macdonald, and I. Ounis. Identifying local events by using microblogs as social sensors. In *10th Conference on Open Research Areas in Information Retrieval (OAIR)*, pp. 173–180. ACM, Paris (France), May 2013. ISBN 978-2-905450-09-8.

[AMRT11] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[AMT05] P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *ACM Symposium on Applied Computing (SAC)*, pp. 1589–1593. ACM, New York, (USA), March 2005. ISBN 1-58113-964-0.

[ANPS11] Y. Azar, A. Nisgav, and B. Patt-Shamir. Recommender systems with non-binary grades. In *23th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pp. 245–252. ACM, New York (USA), June 2011. ISBN 978-1-4503-0743-7.

[Apa05] Apache Software Foundation. Apache Lucene, 2005. `https://lucene.apache.org`, [Accessed on October 3, 2017].

[Apa14] Apache Software Foundation. Apache Mahout 0.9, 2014. `http://mahout.apache.org`, [Accessed on October 3, 2017].

[ARD+16] M. F. Alhamid, M. Rawashdeh, H. Dong, M. A. Hossain, A. Alelaiwi, et al. RecAm: a collaborative context-aware framework for multimedia recommendations in an ambient intelligence environment. *Multimedia Systems*, 22(5):587–601, October 2016. ISSN 1432-1882.

[ARMCM13] V. Ayala-Rivera, P. McDonagh, T. Cerqueus, and L. Murphy. Synthetic data generation using Benerator tool. *CoRR*, abs/1311.3312, 2013.

[ARN08] O. Averjanova, F. Ricci, and Q. N. Nguyen. Map-based interaction with a conversational mobile recommender system. In *Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, pp. 212–218. IEEE, September 2008. ISBN 978-0-7695-3367-4.

[Aro16] S. Arora. Recommendation engines: How Amazon and Netflix are winning the personalization battle, June 2016. `https://www.martechadvisor.com/articles/customer-experience/recommendation-engines-how-amazon-and-netflix-are-winning-the-personalization-battle`, [Accessed on October 3, 2017].

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *20th International Conference on Very Large Data Bases (VLDB)*, pp. 487–499. Morgan Kaufmann, San Francisco (USA), September 1994. ISBN 1-55860-153-8.

[Asa13a] N. Y. Asabere. A framework for context-aware recommendation in mobile social learning. *Advanced Engineering and Applied Sciences: An International Journal*, 3(1):10–16, 2013. ISSN 2320-3927.

[Asa13b] N. Y. Asabere. Towards a viewpoint of context-aware recommender systems (CARS) and services. *International Journal of Computer Science and Telecommunications*, 4(1):10–29, 2013.

[Asg16] N. Asghar. Yelp dataset challenge: Review rating prediction. *CoRR*, abs/1605.05362, May 2016.

[ASM16] S. Asthana, R. Strong, and A. Megahed. HealthAdvisor: Recommendation system for wearable technologies enabling proactive health monitoring. *arXiv preprint arXiv:1612.00800*, abs/1612.00800, 2016.

[ASST05] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, January 2005.

[AT05] G. Adomavicius and A. Tuzhilin. Towards 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, June 2005. ISSN 1041-4347.

[AT08] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *ACM Conference on Recommender Systems (RecSys)*, pp. 335–336. ACM, New York (USA), 2008. ISBN 978-1-60558-093-7.

[AT11] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 7, pp. 217–253. Springer, 2011. ISBN 978-0-387-85820-3.

[AT14] P. Adamopoulos and A. Tuzhilin. Estimating the value of multi-dimensional data sets in context-based recommender systems. In *Poster Proceedings of the 8th ACM Conference on Recommender Systems (RecSys*, volume 1247. CEUR Workshop Proceedings, October 2014. ISSN 1613-0073.

[ATM17] K. Ajudiya, A. Thakkar, and K. Makwana. Recommendation system for improvement in post harvesting of horticulture crops. In S. C. Satapathy and A. Joshi, editors, *Second International Conference on Information and Communication Technology for Intelligent Systems (ICTIS)*, volume 84 of *Smart Innovation, Systems and Technologies (SIST)*, pp. 546–553. Springer, Cham, March 2017. ISBN 978-3-319-63645-0.

[AW14] Y. A. Asikin and W. Wörndl. Stories Around You: Location-based serendipitous recommendation of news articles. In *Second International Workshop on News Recommendation and Analytics (NRA)*, volume 1181, pp. 1–8. CEUR Workshop Proceedings, July 2014. ISSN 1613-0073.

[AZK15] A. Abbas, L. Zhang, and S. U. Khan. A survey on context-aware recommender systems based on computational intelligence techniques. *Computing*, 97(7):667–690, 2015. ISSN 1436-5057.

[BA12] P. Bedi and S. K. Agarwal. A situation-aware proactive recommender system. In *12th International Conference on Hybrid Intelligent Systems (HIS)*, pp. 85–89. IEEE, December 2012. ISBN 978-1-4673-5116-4.

[BASJ14] P. Bedi, S. K. Agarwal, S. Sharma, and H. Joshi. SAPRS: Situation-aware proactive recommender system with explanations. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 277–283. IEEE, September 2014. ISBN 978-1-4799-3080-7.

[BBC+08] V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, et al. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. In *SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 1157–1166. ACM, New York (USA), April 2008. ISBN 978-1-60558-011-1.

[BBG12] D. Bouneffouf, A. Bouzeghoub, and A. L. Gancarski. Following the user's interests in mobile context-aware recommender systems: The hybrid-e-greedy algorithm. In *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 657–662. IEEE, March 2012. ISBN 978-1-4673-0867-0.

[BBK10] M. Böhmer, G. Bauer, and A. Krüger. Exploring the design space of context-aware recommender systems that suggest mobile applications. In *Second Workshop on Context-Aware Recommender Systems (CARS)*, volume 5, pp. 1–5. September 2010.

[BCGS13] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile Ad Hoc networking: the cutting edge directions*, volume 35. John Wiley & Sons, second edition, 2013. ISBN 978-1-118-08728-2.

[BCHC09] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*, volume 2 of *Springer Topics in Signal Processing (STSP)*, chapter 5, pp. 1–4. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-00296-0.

[BCKO15] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR*, abs/1505.03014, May 2015.

[BDG+10] S. Bergamaschi, E. Domnori, F. Guerra, M. Orsini, R. T. Lado, et al. Keymantic: Semantic keyword-based searching in data integration systems. *Proceedings of the VLDB Endowment*, 3(1–2):1637–1640, 2010. ISSN 2150-8097.

[BDH13] W. Beer, C. Derwein, and S. Herramhof. Implementation of context-aware item recommendation through MapReduce data aggregation. In *11th International Conference on Advances in Mobile Computing & Multimedia (MoMM)*, pp. 26–32. ACM, New York (USA), December 2013. ISBN 978-1-4503-2106-8.

[BDR07] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007. ISSN 1743-8225.

[Ber91] V. Bergmann. Databene Benerator, 1991. `http://databene.org/databene-benerator`, `https://sourceforge.net/projects/benerator/`, [Accessed on October 3, 2017].

[BER13a] M. Braunhofer, M. Elahi, and F. Ricci. Context-aware dataset: STS–South Tyrol suggests mobile app data, 2013. `https://www.researchgate.net/publication/305682479_Context-Aware_Dataset_STS_-_South_Tyrol_Suggests_Mobile_App_Data`, [Accessed on October 3, 2017].

[BER13b] M. Braunhofer, M. Elahi, and F. Ricci. South Tyrol Suggests, 2013. `http://people.stern.nyu.edu/padamopo/data/concertTweets.html`, [Accessed on October 3, 2017].

[BER14] M. Braunhofer, M. Elahi, and F. Ricci. STS: A context-aware mobile recommender system for places of interest. In *22nd International Conference on User Modeling, Adaptation, and Personalization (UMAP)-Posters, Demos, Late-breaking Results and Workshop*, pp. 75–80. CEUR Workshop Proceedings, July 2014. ISSN 1613-0073.

[BERS13] M. Braunhofer, M. Elahi, F. Ricci, and T. Schievenin. Context-aware Points of Interest suggestion with dynamic weather data management. In

Z. Xiang and I. Tussyadiah, editors, *International Conference on Information and Communication Technologies in Tourism*, pp. 87–100. Springer, Cham, January 2013. ISBN 978-3-319-03973-2.

[Bez94] J. Bezos. Amazon, 1994. American Electronic Commerce and Cloud Computing Company, `https://www.amazon.com`, [Accessed on October 3, 2017].

[BG04] Y. Bengio and Y. Grandvalet. No unbiased estimator of the variance of K-fold cross-validation. *Journal of Machine Learning Research*, 5(September):1089–1105, 2004.

[BGLB16] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, November 2016. ISSN 1432-1300.

[BGRV11] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. A Hidden Markov Model approach to keyword-based search over relational databases. In M. Jeusfeld, L. Delcambre, and T.-W. Ling, editors, *30th International Conference Conceptual Modeling–ER 2011*, volume 6998 of *Lecture Notes in Computer Science (LNCS)*, pp. 411–420. Springer, Berlin, Heidelberg, October 2011. ISBN 978-3-642-24605-0.

[BGT06] A. Barreto, J. Greenberg, and S. Tarantino. Grooveshark, 2006. `http://grooveshark.com`, [Accessed on October 3, 2017].

[BGT15] A. Barreto, J. Greenberg, and S. Tarantino. Datasets for the evaluation of context-aware recommendation systems, 2015. `https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets`, [Accessed on October 3, 2017].

[BH16] V. Bhatia and V. Hasija. Targeted advertising using behavioural data and social data mining. In *Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 937–942. IEEE, July 2016. ISBN 978-1-4673-9991-3. ISSN 2165-8536.

[BHC98] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *15th International Conference on Artificial Intelligence (AAAI)*, pp. 714–720. 1998.

[BHD13] W. Beer, S. Herramhof, and C. Derwein. Implementation of a Map-Reduce based context-aware recommendation engine for social music events. *International Journal On Advances in Intelligent Systems*, 6(3 and 4):367–375, 2013. ISSN 1942-2679.

[BHE00] N. Bulusu, J. Heidemann, and D. Estrin. GPS–less low–cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, October 2000. ISSN 1070-9916.

[BHHD13] W. Beer, W. Hargassner, S. Herramhof, and C. Derwein. General framework for context-aware recommendation of social events. In *Second International Conference on Intelligent Systems and Applications (INTELLI)*, pp. 141–146. IARIA, April 2013. ISBN 978-1-61208-269-1.

[BHK98] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in*

Artificial Intelligence (UAI), pp. 43–52. Morgan Kaufmann, San Francisco (USA), July 1998. ISBN 1-55860-555-X.

[BKKK16] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur. Opinion mining and sentiment analysis. In *Third International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 452–455. IEEE, March 2016. ISBN 978-9-3805-4421-2.

[BKL+11] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, et al. InCarMusic: Context-aware music recommendations in a car. In C. Huemer and T. Setzer, editors, *12th International Conference on E-Commerce and Web Technologies (EC-Web)*, volume 85 of *Lecture Notes in Business Information Processing (LNBIP)*, pp. 89–100. Springer, August 2011. ISBN 978-3-642-23013-4.

[BKR13] M. Braunhofer, M. Kaminskas, and F. Ricci. Location-aware music recommendation. *International Journal of Multimedia Information Retrieval*, 2(1):31–44, March 2013. ISSN 2192-662X.

[BL07] J. Bennett and S. Lanning. The Netflix Prize. In *KDD Cup and Workshop 2007 (KDDCup)*, pp. 3–6. August 2007.

[BL15] I. Benouaret and D. Lenne. Personalizing the museum experience through context-aware recommendations. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 743–748. IEEE, October 2015. ISBN 978-1-4799-8697-2.

[BLJ+09] N. Balacheff, S. Ludvigsen, T. D. Jong, A. Lazonder, S.-A. Barnes, et al. *Technology–Enhanced Learning*. Springer, 2009. ISBN 978-1-4020-9827-7.

[BLPR12] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012. ISSN 1617-4909.

[BLR11] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 301–304. ACM, New York (USA), October 2011. ISBN 978-1-4503-0683-6.

[BLSH10] S. Berkovsky, E. W. D. Luca, A. Said, and J. Hermanns. The challenge on context-aware movie recommendation. In *Workshop on Context-Aware Movie Recommendation (CAMRa)*. ACM, New York (USA), September 2010. ISBN 978-1-4503-0258-6. Special Issue of ACM Transactions on Intelligent Sy stems and Technology.

[BMT+05] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P search. In *31st International Conference on Very Large Data Bases (VLDB)*, pp. 1263–1266. VLDB Endowment, August 2005. ISBN 1-59593-154-6.

[BN16] S. Bobek and G. J. Nalepa. Uncertainty handling in rule-based mobile context-aware systems. *Pervasive and Mobile Computing*, pp. 1–21, October 2016. ISSN 1574-1192.

[BNCM12] M. J. Barranco, J. M. Noguera, J. Castro, and L. Martínez. A context-aware mobile recommender system based on location and trajectory. In

J. Casillas, F. J. Martínez-López, and J. M. C. Rodríguez, editors, *First International Symposium on Management Intelligent Systems*, volume 171 of *Advances in Intelligent Systems and Computing (AISC)*, pp. 153–162. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-30864-2.

[BNWP11] R. Bader, E. Neufeld, W. Woerndl, and V. Prinz. Context-aware POI recommendations in an automotive scenario using multi-criteria decision making methods. In *Workshop on Context-awareness Retrieval and Recommendation (CaRR)*, pp. 23–30. ACM, New York (USA), February 2011. ISBN 978-1-4503-0625-6.

[BOHB12] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012. ISSN 0950-7051.

[BOHG13] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013. ISSN 0950-7051.

[BP98] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *15th International Conference on Machine Learning (ICML)*, pp. 46–54. Morgan Kaufmann, San Francisco (USA), July 1998. ISBN 1-55860-556-8.

[BP00] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2):147–180, June 2000. ISSN 1573-1391.

[BP12] A. Bilge and H. Polat. An improved privacy-preserving DWT-based collaborative filtering scheme. *Expert Systems with Applications*, 39(3):3841–3854, 2012. ISSN 0957-4174.

[BR09a] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 245–248. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[BR09b] L. Baltrunas and F. Ricci. Context-dependent items generation in collaborative filtering. In *International Workshop on Context-Aware Recommender Systems (CARS)*, pp. 22–25. October 2009.

[BR14] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, 24(1):7–34, 2014. ISSN 1573-1391.

[Bra03] M. Brand. Fast online SVD revisions for lightweight recommender systems. In *SIAM International Conference on Data Mining*, pp. 37–46. SIAM, May 2003. ISBN 978-1-61197-273-3.

[BRLW15] M. Braunhofer, F. Ricci, B. Lamche, and W. Wörndl. A context-aware model for proactive recommender systems in the tourism domain. In *17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI)*, pp. 1070–1075. ACM, New York (USA), August 2015. ISBN 978-1-4503-3653-6.

[Bro16] M. Brocato. Mockaroo, 2016. By Mockaroo, LLC., `https://www.mockaroo.com`, [Accessed on October 3, 2017].

[BSHA08] P. Bellekens, K. V. D. Sluijs, G.-J. Houben, and L. Aroyo. On-the-fly data integration for personalized television recommender systems. In *Eighth International Conference on Web Engineering (ICWE)*, pp. 362–365. IEEE, July 2008. ISBN 978-0-7695-3261-5.

[BSW11] R. Bader, O. Siegmund, and W. Woerndl. A study on user acceptance of proactive in-vehicle recommender systems. In *Third International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI)*, pp. 47–54. ACM, New York (USA), November 2011. ISBN 978-1-4503-1231-8.

[Bur02] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002. ISSN 1573-1391.

[Bur05] R. Burke. Hybrid systems for personalized recommendations. In B. Mobasher and S. S. Anand, editors, *IJCAI 2003 Workshop, ITWP 2003: Intelligent Techniques for Web Personalization (ITWP)*, volume 3169 of *Lecture Notes in Computer Science (LNCS)*, pp. 133–152. Springer, Berlin, Heidelberg, August 2005. ISBN 978-3-540-31655-8.

[Bur07] R. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science (LNCS)*, chapter 12, pp. 377–408. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, volume 463. ACM Press New York, 1999. ISBN 0-201-39829-X.

[BZ14] R. Benlamri and X. Zhang. Context-aware recommender for mobile learners. *Human-centric Computing and Information Sciences*, 4(1):1–34, August 2014. ISSN 2192-1962.

[BZB+08] F. Bohnert, I. Zukerman, S. Berkovsky, T. Baldwin, and L. Sonenberg. Using interest and transition models to predict visitor locations in museums. *AI Communications*, 21(2–3):195–202, 2008.

[BZM12] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pp. 199–208. ACM, New York (USA), November 2012. ISBN 978-1-4503-1691-0.

[BZWM15] J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel. A survey on recommendations in location-based social networks. *GeoInformatica*, 19(3):525–565, 2015.

[CB05] Y. Cui and S. Bull. Context and learner modelling for the mobile foreign language learner. *System*, 33(2):353–367, 2005. ISSN 0346-251X.

[CBC08a] I. Cantador, A. Bellogín, and P. Castells. News@hand: A Semantic Web approach to recommending news. In W. Nejdl, J. Kay, P. Pu, and E. Herder, editors, *Fifth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, volume 5149 of *Lecture Notes in Computer Science (LNCS)*, pp. 279–283. Springer, Berlin, Heidelberg, July 2008.

[CBC08b] I. Cantador, A. Bellogín, and P. Castells. Ontology-based personalised and context-aware recommendations of news items. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 1, pp. 562–565. IEEE, Washington (USA), December 2008. ISBN 978-0-7695-3496-1.

[CBJC11] B. Chihani, E. Bertin, F. Jeanne, and N. Crespi. Context-aware systems: A case study. In H. Cherifi, J. M. Zain, and E. El-Qawasmeh, editors, *International Conference on Digital Information and Communication Technology and Its Applications (DICTAP)*, volume 167 of *Communications in Computer and Information Science (CCIS)*, pp. 718–732. Springer, Berlin, Heidelberg, June 2011. ISBN 978-3-642-22027-2.

[CC08] G.-D. Chen and P.-Y. Chao. Augmenting traditional books with context-aware learning supports from online learning communities. *Journal of Educational Technology & Society*, 11(2):27–40, April 2008.

[CC12] A. Campbell and T. Choudhury. From smart to cognitive phones. *IEEE Pervasive Computing*, 11(3):7–11, 2012. ISSN 1536-1268.

[CCFF11] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2:1–2:33, February 2011. ISSN 1559-1131.

[CCL13] I. Chlamtac, M. Conti, and J. J.-N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, 2013. ISSN 1570-8705.

[CCYX09] H. Cao, E. Chen, J. Yang, and H. Xiong. Enhancing recommender systems under volatile userinterest drifts. In *18th ACM Conference on Information and Knowledge Management (CIKM)*, pp. 1257–1266. ACM, New York (USA), November 2009. ISBN 978-1-60558-512-3.

[CCZ17] C. Chen, K. C.-C. Chang, and X. Zheng. Towards context-aware social recommendation via individual trust. *Knowledge-Based Systems*, pp. 1–9, March 2017. ISSN 0950-7051.

[CD14] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)?–arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[CDC14] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1):67–119, 2014. ISSN 1573-1391.

[CDI11] N. Cenerario, T. Delot, and S. Ilarri. A content-based dissemination protocol for VANETs: Exploiting the encounter probability. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):771–782, September 2011. ISSN 1524-9050.

[CEL+06] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *Second Annual International Workshop on Wireless Internet (WICON)*, pp. 1–14. ACM, New York (USA), August 2006. ISBN 1-59593-510-X.

[CFLHRM10] L. M. D. Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, September 2010. ISSN 0888-613X.

[CFP+12] H. Costa, B. Furtado, D. Pires, L. Macedo, and A. Cardoso. Context and intention-awareness in POIs recommender systems. In *Fourth Workshop on Context-Aware Recommender Systems (CARS), in conjunction with the Sixth ACM Conference on Recommender Systems (RecSys)*, volume 12. September 2012. ISBN 978-1-4503-0683-6.

[CFTCD13] P. G. Campos, I. Fernández-Tobías, I. Cantador, and F. Díez. Context-aware movie recommendations: An empirical comparison of pre-filtering, post-filtering and contextual modeling approaches. In C. Huemer and P. Lops, editors, *14th International Conference E-Commerce and Web Technologies (EC-Web)*, pp. 137–149. Springer, Berlin, Heidelberg, August 2013. ISBN 978-3-642-39878-0.

[CG14] M. Conti and S. Giordano. Mobile ad hoc networking: milestones, challenges, and new research directions. *IEEE Communications Magazine*, 52(1):85–96, January 2014. ISSN 0163-6804.

[CGI+15] S. Cadegnani, F. Guerra, S. Ilarri, M. del Carmen Rodríguez-Hernández, R. Trillo-Lado, et al. Recommending web pages using item-based collaborative filtering approaches. In J. Cardoso, F. Guerra, G.-J. Houben, A. M. Pinto, and Y. Velegrakis, editors, *Semantic Keyword-based Search on Structured Data Sources: First COST Action IC1302 International KEYSTONE Conference (IKC)*, volume 9398 of *Lecture Notes in Computer Science (LNCS)*, pp. 17–29. Springer, September 2015. ISBN 978-3-319-27932-9.

[CGI+17] S. Cadegnani, F. Guerra, S. Ilarri, M. del Carmen Rodríguez-Hernández, R. Trillo-Lado, et al. Exploiting linguistic analysis on URLs for recommending web pages: a comparative study. In N. T. Nguyen, R. Kowalczyk, A. M. Pinto, and J. Cardoso, editors, *Transactions on Computational Collective Intelligence XXVI*, volume 10190 of *Lecture Notes in Computer Science (LNCS)*, pp. 26–45. Springer, Cham, 2017. ISBN 978-3-319-59268-8.

[CGQT11] P. Cremonesi, P. Garza, E. Quintarelli, and R. Turrin. Top-N recommendations on unpopular items with contextual knowledge. In *Third Workshop on Context-aware Recommender Systems (CARS), in conjunction with the Fifth ACM Conference on Recommender Systems (RecSys)*. CEUR Workshop Proceedings, October 2011. ISSN 1613-0073.

[Cha16] C. Chao. JGraphX, 2016. `https://github.com/jgraph/jgraphx`, [Accessed on October 3, 2017].

[Che05a] A. Chen. Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. In T. Strang and C. Linnhoff-Popien, editors, *First International Workshop on Location– and Context-Awareness (LoCA)*, volume 3479 of Lecture Notes in Computer Science, pp. 244–253. Springer, May 2005.

[Che05b] A. Chen. Context-aware collaborative filtering system: Predicting the user's preferences in ubiquitous computing. In *Extended Abstracts on Hu-*

man Factors in Computing Systems (CHI), pp. 1110–1111. ACM, New York (USA), April 2005. ISBN 1-59593-002-7.

[CHGS05] S.-C. Chou, W.-T. Hsieh, F. L. Gandon, and N. M. Sadeh. Semantic Web technologies for context-aware museum tour guide applications. In *19th International Conference on Advanced Information Networking and Applications (AINA)*, volume 2, pp. 709–714. IEEE, March 2005. ISBN 0-7695-2249-1. ISSN 2332-5658.

[CHPY15] C.-C. Chen, T.-C. Huang, J. J. Park, and N. Y. Yen. Real-time smartphone sensing and recommendations towards context-awareness shopping. *Multimedia systems*, 21(1):61–72, February 2015.

[CHY+17] L. Cui, W. Huang, Q. Yan, F. R. Yu, Z. Wen, et al. A novel context-aware recommendation algorithm with two-level SVD in social networks. *Future Generation Computer Systems*, July 2017. To appear.

[CK00] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Deptartment of Computer Science, Hanover (USA), 2000.

[CK14] E. Christakopoulou and G. Karypis. HOSLIM: Higher-Order Sparse Linear Method for top-N recommender systems. In V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen, and H.-Y. Kao, editors, *18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 8444 of *Lecture Notes in Computer Science (LNCS)*, pp. 38–49. Springer, Cham, May 2014. ISBN 978-3-319-06605-9.

[CKY08] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *25th International Conference on Machine Learning (ICML)*, pp. 96–103. ACM, New York (USA), July 2008. ISBN 978-1-60558-205-4.

[CL14] K.-C. Chen and S.-Y. Lien. Machine-to-machine communications: Technologies and challenges. *Ad Hoc Networks*, 18:3–23, July 2014. ISSN 1570-8705.

[CLG+10] L. Cao, J. Luo, A. Gallagher, X. Jin, J. Han, et al. Aworldwide tourism recommendation system based on geotaggedweb photos. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 2274–2277. IEEE, March 2010. ISBN 978-1-4244-4295-9. ISSN 1520-6149.

[CLLP17] F. Clarizia, S. Lemma, M. Lombardi, and F. Pascale. A mobile context-aware information system to support tourism events. In M. H. A. Au, A. Castiglione, K.-K. R. Choo, F. Palmieri, and K.-C. Li, editors, *12th International Conference on Green, Pervasive, and Cloud Computing (GPC)*, volume 10232 of *Lecture Notes in Computer Science (LNCS)*, pp. 553–566. Springer, May 2017. ISBN 978-3-319-57186-7.

[CMBMRL+11] E. Costa-Montenegro, A. B. Barragáns-Martínez, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro-Ramallo. Which App? A recommender system of applications in markets by monitoring users' interaction. In *2011 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 353–354. IEEE, January 2011. ISBN 978-1-4244-8712-7. ISSN 2158-3994.

[CMBMRL12] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López. Which App? a recommender system of applications in markets: Implementation of the service for monitoring users interaction. *Expert Systems with Applications*, 39(10):9367–9375, August 2012. ISSN 0957-4174.

[CMGSS13] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE Wireless Communications*, 20(3):96–104, June 2013. ISSN 1536-1284.

[CMMP13] A. Chianese, F. Marulli, V. Moscato, and F. Piccialli. SmARTweet: A location-based smart application for exhibits and museums. In K. Yetongnon, A. Dipanda, and R. Chbeir, editors, *International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pp. 408–415. IEEE, December 2013. ISBN 978-1-4799-3211-5.

[CMS15] C. Campolo, A. Molinaro, and R. Scopigno. *Vehicular Ad Hoc Networks: Standards, Solutions, and Research*. Springer, first edition, 2015. ISBN 978-3-319-15497-8.

[CMT+08] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, et al. Activity sensing in the wild: a field trial of ubifit garden. In *SIGCHI Intenrational Conference on Human Factors in Computing Systems (CHI)*, pp. 1797–1806. ACM, New York (USA), April 2008. ISBN 978-1-60558-011-1.

[CMVGRG+15] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater. RecomMetz: A context-aware knowledge-based mobile recommender system for movie showtimes. *Expert Systems with Applications*, 42(3):1202–1222, 2015. ISSN 0957-4174.

[CNL+12] D. Cao, W. Nie, R. Li, Z. Li, T. Wang, et al. Context-aware recommendation in mobile environments: An approach based on interest resonance. *Wuhan University Journal of Natural Sciences*, 17(5):400–406, October 2012. ISSN 1993-4998.

[Coa04] S. Coast. OpenStreetMap, 2004. `https://www.openstreetmap.org`, [Accessed on October 3, 2017].

[COO13] V. Coskun, B. Ozdenizci, and K. Ok. A survey on near field communication (NFC) technology. *Wireless Personal Communications*, 71(3):2259–2294, August 2013. ISSN 1572-834X.

[Cor16] Corban Works, LLC. Fake Name Generator, 2016. `http://www.fakenamegenerator.com`, [Accessed on October 3, 2017].

[CPGPSM16] R. Colomo-Palacios, F. J. García-Peñalvo, V. Stantchev, and S. Misra. Towards a social and context-aware mobile recommendation system for tourism. *Pervasive and Mobile Computing*, 2016. ISSN 1574-1192.

[CRB+03] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 407–418. ACM, New York (USA), August 2003. ISBN 1-58113-735-4.

[CRC13a] V. Codina, F. Ricci, and L. Ceccaroni. Exploiting the semantic similarity of contextual situations for pre-filtering recommendation. In S. Carberry,

S. Weibelzahl, A. Micarelli, and G. Semeraro, editors, *21th International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, volume 7899 of *Lecture Notes in Computer Science (LNCS)*, pp. 165–177. Springer, Berlin, Heidelberg, June 2013. ISBN 978-3-642-38844-6.

[CRC13b] V. Codina, F. Ricci, and L. Ceccaroni. Local context modeling with semantic pre-filtering. In *Seventh ACM Conference on Recommender Systems (RecSys)*, pp. 363–366. ACM, New York (USA), October 2013. ISBN 978-1-4503-2409-0.

[CRC16] V. Codina, F. Ricci, and L. Ceccaroni. Distributional semantic pre-filtering in context-aware recommender systems. *User Modeling and User-Adapted Interaction*, 26(1):1–32, March 2016. ISSN 1573-1391.

[CS09] D. Crowley and N. Selvadurai. Foursquare, 2009. `http://foursquare.com`, [Accessed on October 3, 2017].

[CS14] Z. Cheng and J. Shen. Just-for-Me: An adaptive personalization system for location-aware social music recommendation. In *International Conference on Multimedia Retrieval (ICMR)*, pp. 185:185–185:192. ACM, New York (USA), April 2014. ISBN 978-1-4503-2782-4.

[CSdVR11] M. Clements, P. Serdyukov, A. P. de Vries, and M. J. T. Reinders. Personalised travel recommendation based on location co-occurrence. *CoRR*, abs/1106.5213:1–30, 2011.

[CSS10] S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Engineering Bulletin*, 33(1):3–24, 2010.

[CSS15a] Z. D. Champiri, S. S. B. Salim, and S. R. Shahamiri. The role of context for recommendations in digital libraries. *International Journal of Social Science and Humanity*, 5(11):948–954, 2015.

[CSS15b] Z. D. Champiri, S. R. Shahamiri, and S. S. B. Salim. A systematic review of scholar context-aware recommender systems. *Expert Systems with Applications*, 42(3):1743–1758, February 2015. ISSN 0957-4174.

[CT03] G. C. Cawley and N. L. C. Talbot. Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. *Pattern Recognition*, 36(11):2585–2592, November 2003. ISSN 0031-3203.

[CT04] G. C. Cawley and N. L. C. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17(10):1467–1475, December 2004. ISSN 0893-6080.

[CZC+15] X. Chen, Y. Zeng, G. Cong, S. Qin, Y. Xiang, et al. On information coverage for location category based Point-of-interest recommendation. In *29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 37–43. AAAI Press, January 2015. ISBN 0-262-51129-0.

[CZW+07] R. Cai, C. Zhang, C. Wang, L. Zhang, and W.-Y. Ma. MusicSense: Contextual music recommendation using emotional allocation modeling. In *15th ACM International Conference on Multimedia (MM)*, pp. 553–556. ACM, New York (USA), September 2007. ISBN 978-1-59593-702-5.

[CZZC13] C. Chen, J. Zeng, X. Zheng, and D. Chen. Recommender system based on social trust relationships. In *10th International Conference on e-Business*

Engineering (ICEBE), pp. 32–37. IEEE, September 2013. ISBN 978-0-7695-5111-1.

[dCPHSS12] J. V. del Campo, J. Pegueroles, J. Hernández-Serrano, and M. Soriano. Design of a P2P content recommendation system using affinity networks. *Computer Communications*, 36(1):90–104, December 2012. ISSN 0140-3664.

[dCRHGITL15] M. del Carmen Rodríguez-Hernández, F. Guerra, S. Ilarri, and R. Trillo-Lado. A first step towards keyword-based searching for recommendation systems. In *XX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*. September 2015. 4 pp., `http://hdl.handle.net/11705/JISBD/2015/007`.

[dCRHI14a] M. del Carmen Rodríguez-Hernández and S. Ilarri. Context-aware recommendations in mobile environments. In J. Tuya, M. Ruiz, and N. Hurtado, editors, *XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pp. 85–90. September 2014. ISBN 978-84-697-1152-1, 84-697-1152-0.

[dCRHI14b] M. del Carmen Rodríguez-Hernández and S. Ilarri. Researching context-aware recommendation systems in mobile environments. *Actas de las III Jornadas de Jóvenes Investigadores del I3A*, 2:59–60, June 2014. ISSN 2341-4790.

[dCRHI14c] M. del Carmen Rodríguez-Hernández and S. Ilarri. Towards a context-aware mobile recommendation architecture. In I. Awan, M. Younas, X. Franch, and C. Quer, editors, *11th International Conference on Mobile Web Information Systems (MobiWIS)*, volume 8640 of *Lecture Notes in Computer Science (LNCS)*, pp. 56–70. Springer, Cham, August 2014. ISBN 978-3-319-10359-4.

[dCRHI16] M. del Carmen Rodríguez-Hernández and S. Ilarri. Pull-based recommendations in mobile environments. *Computer Standards & Interfaces*, 44:185–204, February 2016. ISSN 0920-5489.

[dCRHIHTL17a] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. DataGenCARS: A generator of synthetic data for the evaluation of context-aware recommendation systems. *Pervasive and Mobile Computing*, 38, Part 2(2017):516–541, July 2017. ISSN 1574-1192. Special Issue on Context-aware Mobile Recommender Systems.

[dCRHIHTL17b] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. Definiendo un caso de estudio para recomendaciones dinámicas móviles. In *XXII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*. July 2017. 4 pp., `http://hdl.handle.net/11705/JISBD/2017/013`.

[dCRHIHTL17c] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Hermoso, and R. Trillo-Lado. Towards trajectory-based recommendations in museums: Evaluation of strategies using mixed synthetic and real data. In *Eighth International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN)*, volume 113, pp. 234–239. September 2017. ISSN 1877-0509. Procedia Computer Science.

[dCRHITH17] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo, and R. Hermoso. Context-aware recommendations using mobile P2P. In *15th International Conference on Advances in Mobile Computing and Multimedia (MoMM)*. ACM, New York (USA), December 2017. ISBN 978-1-4503-5300-7. 10 pp., accepted, to appear.

[dCRHITLG16] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo-Lado, and F. Guerra. Towards keyword-based pull recommendation systems. In *18th International Conference on Enterprise Information Systems (ICEIS)*, volume 1, pp. 207–214. SCITEPRESS (Science and Technology Publications, Lda.), April 2016. ISBN 978-989-758-187-8.

[dCRHITLH15] M. del Carmen Rodríguez-Hernández, S. Ilarri, R. Trillo-Lado, and R. Hermoso. Location-aware recommendation systems: Where we are and where we recommend to go. In *International Workshop on Location-Aware Recommendations (LocalRec), in conjunction with the Ninth ACM Conference on Recommender Systems (RecSys)*, volume 1405, pp. 1–8. CEUR Workshop Proceedings, September 2015. ISSN 1613-0073.

[DD10] G. Dimitrakopoulos and P. Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, March 2010. ISSN 1556-6072.

[DDGR07] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google News personalization: Scalable online collaborative filtering. In *16th International Conference on World Wide Web (WWW)*, pp. 271–280. ACM, New York (USA), May 2007. ISBN 978-1-59593-654-7.

[Dey01] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, February 2001. ISSN 1617-4909.

[DG08] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, January 2008. ISSN 0001-0782.

[DGB09] W. Dubitzky, M. Granzow, and D. P. Berrar. *Fundamentals of Data Mining in Genomics and Proteomics*. Springer, Berlin, Heidelberg, 2009. ISBN 978-0-387-47509-7.

[DGK17] K. Dou, B. Guo, and L. Kuang. A privacy-preserving multimedia recommendation in the context of social network based on weighted noise injection. *Multimedia Tools and Applications*, pp. 1–20, 2017. ISSN 1573-7721.

[DGMS16] D. D'Agostino, F. Gasparetti, A. Micarelli, and G. Sansonetti. A social context-aware recommender of itineraries between relevant Points of Interest. In C. Stephanidis, editor, *18th International Conference on Human-Computer Interaction (HCI)*, Communications in Computer and Information Science (CCIS), pp. 354–359. Springer, Cham, July 2016. ISBN 978-3-319-40542-1.

[DGY+14] R. Duan, R. S. M. Goh, F. Yang, Y. K. Tan, and J. F. B. Valenzuela. Towards building and evaluating a personalized location-based recommender system. In *IEEE International Conference on Big Data (Big Data)*, pp. 43–48. IEEE, October 2014. ISBN 978-1-4799-5666-1.

[DHS00] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000. ISBN 0471056693.

[DKH+98] B. J. Dahlen, J. A. Konstan, J. L. Herlocker, N. Good, A. Borchers, et al. Jump-starting movieLens: User benefits of starting a collaborative filtering system with "dead data". techreport 98-017, University of Minnesota, March 1998.

[DLL+10] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. V. Vleet, et al. The YouTube video recommendation system. In *Fourth ACM Conference on Recommender Systems (RecSys)*, pp. 293–296. ACM, New York (USA), September 2010. ISBN 978-1-60558-906-0.

[DLNW13] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, december 2013.

[DMG12] J. Das, S. Majumder, and P. Gupta. Voronoi based location aware collaborative filtering. In *Third National Conference on Emerging Trends and Applications in Computer Science (NCETACS)*, pp. 179–183. IEEE, March 2012. ISBN 978-1-4577-0748-3.

[dMNCR14] A. R. de M. Neves, Á. M. G. Carvalho, and C. G. Ralha. Agent-based architecture for context-aware and personalized event recommendation. *Expert Systems with Applications*, 41(2):563–573, February 2014. ISSN 0957-4174.

[dOG08] F. H. del Olmo and E. Gaudioso. Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, 35(3):790–804, October 2008. ISSN 0957-4174.

[Dor07] J. Dorsey. Twitter, 2007. `https://twitter.com`, [Accessed on October 3, 2017].

[Dou04] P. Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, 2004. ISSN 1617-4917.

[DP97] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, November 1997. ISSN 1573-0565. Special issue on learning with probabilistic representations.

[DP16] P. Davidson and R. Piche. A survey of selected indoor positioning methods for smartphones. *IEEE Communications Surveys & Tutorials*, PP(99):1–44, December 2016. ISSN 1553-877X.

[DPK11] F. Draidi, E. Pacitti, and B. Kemme. P2Prec: A P2P recommendation system for large-scale data sharing. In A. Hameurlain, J. Küng, and R. Wagner, editors, *Transactions on Large-Scale Data– and Knowledge-Centered Systems III: Special Issue on Data and Knowledge Management in Grid and P2P Systems*, volume 6790 of *Lecture Notes in Computer Science (LNCS)*, pp. 87–116. Springer, Berlin, Heidelberg, 2011. ISBN 978-3-642-23074-5.

[DPPV11] F. Draidi, E. Pacitti, D. Parigot, and G. Verger. P2Prec: A social-based P2P recommendation system. In *20th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 2593–2596. ACM, New York (USA), October 2011. ISBN 978-1-4503-0717-8.

[Dru17] C. Drummond. En 2021 habrá más personas con smartphone que con agua corriente, 2017. `http://w.ticbeat.com/vakws3`, [Accessed on October 3, 2017].

[dSAB12] F. S. da Silva, L. G. P. Alves, and G. Bressan. Personal TVware: an infrastructure to support the context-aware recommendation for personalized digital TV. *International Journal of Computer Theory and Engineering*, 4(2):131–136, April 2012.

[DTB12] A. Doryab, J. Togelius, and J. Bardram. Activity-aware recommendation for collaborative work in operating rooms. In *17th ACM International Conference on Intelligent User Interfaces (IUI)*, pp. 301–304. ACM, New York (USA), February 2012. ISBN 978-1-4503-1048-2.

[DVR17] K. Dai, A. F. Vilas, and R. P. D. Redondo. A new MOOCs' recommendation framework based on LinkedIn data. In E. Popescu, Kinshuk, M. K. Khribi, R. Huang, M. Jemni, et al., editors, *Innovations in Smart Learning*, Lecture Notes in Educational Technology (LNET), pp. 19–22. Springer, Singapore, September 2017. ISBN 978-981-10-2419-1.

[DYS+17] S. Demirci, A. Yardimci, M. Sayit, E. T. Tunali, and H. Bulut. A hierarchical P2P clustering framework for video streaming systems. *Computer Standards & Interfaces*, 49:44–58, January 2017. ISSN 0920-5489.

[EBOY07] M. M. El-Bishouty, H. Ogata, and Y. Yano. PERKAM: Personalized knowledge awareness map for computer supported ubiquitous learning. *Journal of Educational Technology & Society*, 10(3):122–134, July 2007. ISSN 14364522.

[EBRT13] M. Elahi, M. Braunhofer, F. Ricci, and M. Tkalcic. Personality-based active learning for collaborative filtering recommender systems. In M. Baldoni, C. Baroglio, G. Boella, and R. Micalizio, editors, *XIIIth International Conference of the Italian Association for Artificial Intelligence (AI*IA): Advances in Artificial Intelligence*, volume 8249 of *Lecture Notes in Computer Science (LNCS)*, pp. 360–371. Springer, Cham, December 2013. ISBN 978-3-319-03524-6.

[Edd96] S. R. Eddy. Hidden Markov Models. *Current Opinion in Structural Biology*, 6(3):361–365, June 1996. ISSN 0959-440X.

[Eri15] Ericsson Corporate Communications. Ericsson mobility report: 70 percent of world's population using smartphones by 2020, 2015. `https://www.ericsson.com/news/1925907`, [Accessed on October 3, 2017].

[ERK11] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011. ISSN 1551-3955.

[Eth14] D. Etherington. Beats music mobile app gets recommendation tweaking, verified profiles and more, 2014. `https://techcrunch.com/2014/07/28/beats-music-mobile-app-gets-recommendation-tweaking-verified-profiles-and-more`, [Accessed on October 3, 2017].

[EVMT12] S. Ebrahimi, N. M. Villegas, H. A. Müller, and A. Thomo. SmarterDeals: A context-aware deal recommendation system based on the smartercontext

engine. In *2012 Conference of the Center for Advanced Studies on Collaborative Research (CASCON)*, pp. 116–130. IBM Corp., Riverton (USA), November 2012.

[EW17] M. R. Ebling and R. Want. Satya revisits "Pervasive computing: Vision and challenges". *IEEE Pervasive Computing*, 16(3):20–23, 2017. ISSN 1536-1268.

[FBB10] M. Frank, J. M. Buhmann, and D. Basin. On the definition of role mining. In *15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pp. 35–44. ACM, New York (USA), June 2010. ISBN 978-1-4503-0049-0.

[FBY92] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall PTR, 1992. ISBN 0134638379.

[FC17] S. Feng and J. Cao. Improving group recommendations via detecting comprehensive correlative information. *Multimedia Tools and Applications*, 76(1):1355–1377, 2017. ISSN 1573-7721.

[FCF+14] A. Fortier, C. Challiol, J. L. Fernández, S. Robles, G. Rossi, et al. Exploiting personal web servers for mobile context-aware applications. *The Knowledge Engineering Review*, 29(02):134–153, 2014.

[FFF+06] S. Frattasi, H. Fathi, F. H. Fitzek, R. Prasad, and M. D. Katz. Defining 4G technology from the users perspective. *IEEE network*, 20(1):35–41, January 2006. ISSN 0890-8044.

[Fin10] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. John Wiley & Sons, third edition, 2010. ISBN 978-0-470-69506-7.

[Fin14] G. A. Fink. *Hidden Markov Models*, chapter 5, pp. 71–106. Springer, second edition, 2014. ISBN 978-1-4471-6308-4.

[Fis88] R. A. Fisher. Iris Plants Dataset, 1988. `https://archive.ics.uci.edu/ml/datasets/iris`, [Accessed on October 3, 2017].

[FJN+13] A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, and S. Reiterer. Toward the next generation of recommender systems: Applications and research challenges. In G. A. Tsihrintzis, M. Virvou, and L. C. Jain, editors, *Multimedia Services in Intelligent Environments: Advances in Recommender Systems*, volume 24 of *Smart Innovation, Systems and Technologies (SIST)*, pp. 81–98. Springer, 2013. ISBN 978-3-319-00372-6.

[FKS03] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[Fle13] B. Fleming. Sensors – A forecast [Automotive Electronics]. *IEEE Vehicular Technology Magazine*, 8(3):4–12, 2013. ISSN 1556-6072.

[Fou13] Foursquare. Foursquare Dataset, 2013. `https://developer.foursquare.com`, [Accessed on October 3, 2017].

[Fox01] G. Fox. Peer-to-peer networks. *Computing in Science Engineering*, 3(3):75–77, May 2001. ISSN 1521-9615.

[FP00] J. Frankel and T. Pepper. Gnutella, 2000. `http://gnutella.wego.com`, [Accessed on October 3, 2017].

[FPH+16] Z.-W. Fan, Y.-Z. Peng, P.-J. Huang, P.-Y. Lin, Y.-K. Chen, et al. Empirical evaluation of contextual combinations in recommendation system. In *International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pp. 720–725. IEEE, July 2016. ISBN 978-1-5090-0390-7. ISSN 2160-1348.

[FS11] D. C.-L. Fong and M. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

[FZ94] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, April 1994. ISSN 0018-9162.

[GBAH12] D. Gallego, E. Barra, S. Aguirre, and G. Huecas. A model for generating proactive context-aware recommendations in e-learning systems. In *Frontiers in Education Conference (FIE)*, pp. 1–6. IEEE, October 2012. ISBN 978-1-4673-1352-0. ISSN 2377-634X.

[GBMP13] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. ISSN 0167-739X.

[GC13] A. Germain and J. Chakareski. Spotify Me: Facebook-assisted automatic playlist generation. In *15th International Workshop on Multimedia Signal Processing (MMSP)*, pp. 25–28. IEEE, September 2013. ISBN 978-1-4799-0125-8.

[GCM11] G. Gheorghe, R. L. Cigno, and A. Montresor. Security and privacy issues in P2P streaming systems: A survey. *Peer-to-Peer Networking and Applications*, 4(2):75–91, June 2011. ISSN 1936-6450.

[GCX+07] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, et al. A performance study of BitTorrent-like peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 25(1):155–169, January 2007. ISSN 0733-8716.

[GDF73] J. G. David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973. ISSN 0018-9219.

[GDF13] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *Seventh ACM Conference on Recommender Systems (RecSys)*, pp. 105–112. ACM, New York (USA), October 2013. ISBN 978-1-4503-2409-0.

[GF13] F. Garcin and B. Faltings. PEN recsys: a personalized news recommender systems framework. In *International News Recommender Systems Workshop and Challenge (NRS)*, pp. 3–9. ACM, New York (USA), October 2013. ISBN 978-1-4503-2302-4.

[GFD+14] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, et al. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Eighth ACM Conference on Recommender Systems (RecSys)*, pp. 169–176. ACM, New York (USA), October 2014. ISBN 978-1-4503-2668-1.

[GH12] D. Gallego and G. Huecas. An empirical case of a context-aware mobile recommender system in a banking environment. In *Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC)*, pp. 13–20. IEEE, June 2012. ISBN 978-0-7695-4727-5.

[GK12] T. G.S. and U. P. Kulkarni. Design and implementation of user context aware recommendation engine for mobile using Bayesian network, Fuzzy Logic and rule base. *International Journal of Pervasive Computing and Communications*, 8(2):133–157, 2012.

[GK17] C. Golian and J. Kuchar. News recommender system based on association rules at CLEF NewsREEL 2017. In *8th International Conference of the CLEF Initiative*, volume 1866. CEUR Workshop Proceedings, September 2017. ISSN 1613-0073.

[GKMP14] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou. Mobile recommender systems in tourism. *Journal of Network and Computer Applications*, 39:319–333, 2014. ISSN 1084-8045.

[GLX+11] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 983–991. ACM, New York (USA), August 2011. ISBN 978-1-4503-0813-7.

[GM05] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM)*. IEEE, November 2005. ISBN 0-7695-2278-5. ISSN 1550-4786.

[Goo03] Google. Google AdSense, 2003. `http://www.google.com/adsense`, [Accessed on October 3, 2017].

[Goo07] M. F. Goodchild. Citizens as sensors: The world of volunteered geography. *GeoJournal*, 69(4):211–221, August 2007.

[Goo12a] Google. Google Now, 2012. `https://www.google.com/intl/es/landing/now`, [Accessed on October 3, 2017].

[Goo12b] Google. Google Play, 2012. `https://play.google.com/store`, [Accessed on October 3, 2017].

[GPSS09] G. Ghiani, F. Paternò, C. Santoro, and L. D. Spano. UbiCicero: A location-aware, multi-device museum guide. *Interacting with Computers*, 21(4):288–303, June 2009.

[GPT11] M. Gorgoglione, U. Panniello, and A. Tuzhilin. The effect of context-aware recommendations on customer purchasing behavior and trust. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 85–92. ACM, New York (USA), October 2011. ISBN 978-1-4503-0683-6.

[GR70] G. H. Golub and C. Reinsch. Singular Value Decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970. ISSN 0945-3245.

[GRGP01] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, July 2001. ISSN 1573-7659.

[Gro96] GroupLens Research. MovieLens, 1996. `https://movielens.org`, [Accessed on October 3, 2017].

[Gro16] GroupLens Research. MovieLens Dataset, 2016. `https://grouplens.org/datasets/movielens`, [Accessed on October 3, 2017].

[GS09] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(December):2935–2962, 2009.

[GS13] A. Gupta and K. Singh. Location based personalized restaurant recommendation system for mobile environments. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 507–511. IEEE, August 2013. ISBN 978-1-4673-6217-7.

[GU00] H. G. Gök and Ö. Ulusoy. Transmission of continuous query results in mobile computing systems. *Information Sciences*, 125(1-4):37–63, June 2000. ISSN 0020-0255.

[GUH16] C. A. Gomez-Uribe and N. Hunt. The Netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, December 2016. ISSN 2158-656X.

[Guo14] G. Guo. LibRec, 2014. `https://www.librec.net`, [Accessed on October 3, 2017].

[GWH13] D. Gallego, W. Woerndl, and G. Huecas. Evaluating the impact of proactivity in the user experience of a context-aware restaurant recommender for Android smartphones. *Journal of Systems Architecture*, 59(9):748–758, 2013. ISSN 1383-7621.

[GYL11] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, November 2011. ISSN 0163-6804.

[GYT09] S. Gong, H. Ye, and H. Tan. Combining memory-based and model-based collaborative filtering in recommender system. In *Pacific-Asia Conference on Circuits, Communications and Systems (PACCS)*, pp. 690–693. IEEE, May 2009. ISBN 978-0-7695-3614-9.

[GŽB+14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):1–37, March 2014. ISSN 0360-0300.

[HAY+05] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology: Coding and Computing (ITCC)*, volume 2, pp. 205–213. IEEE, April 2005. ISBN 0-7695-2315-3.

[HBG+02] R. Hoffman, A. Blue, K. Guericke, E. Ly, and J.-L. Vaillant. LinkedIn, 2002. `https://www.linkedin.com`, [Accessed on October 3, 2017].

[HBOG17] A. Hernando, J. Bobadilla, F. Ortega, and A. Gutiérrez. A probabilistic model for recommending to new cold-start non-registered users. *Information Sciences*, 376:216–232, 2017. ISSN 0020-0255.

[HC04a] A. Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. In *12th IEEE International Workshop on Quality of Service (IWQOS)*, pp. 171–180. IEEE, June 2004. ISBN 0-7803-8277-3.

[HC04b] C. Hayes and P. Cunningham. Context boosting collaborative recommendations. *Knowledge-Based Systems*, 17(2):131–138, May 2004. ISSN 0950-7051.

[HCC+16] L. Hu, L. Cao, J. Cao, Z. Gu, G. Xu, et al. Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains. *ACM Transactions on Information Systems (TOIS)*, 35(2):13:1–13:37, December 2016. ISSN 1046-8188.

[HCK05] C. Hurley, S. Chen, and J. Karim. YouTube, 2005. `https://www.youtube.com`, [Accessed on October 3, 2017].

[HCXZ14] C. Hu, M. Chen, C. Xing, and G. Zhang. Exploring the optimal substream scheduling and distribution mechanism for data-driven P2P media streaming. *Computer Communications*, 44:14–25, May 2014. ISSN 0140-3664.

[HCZ04] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, January 2004. ISSN 1046-8188.

[Het12] J. T. Hetzel. trapezoid: The trapezoidal distribution, 2012. Package for the R languager, `https://cran.r-project.org/web/packages/trapezoid/vignettes/trapezoid.pdf`, `https://cran.r-project.org/web/packages/trapezoid/trapezoid.pdf`.

[HF08] A. E. Howe and R. D. Forbes. Re-considering neighborhood-based collaborative filtering parameters in the context of new data. In *17th ACM Conference on Information and Knowledge Management (CIKM)*, pp. 1481–1482. ACM, New York (USA), October 2008. ISBN 978-1-59593-991-3.

[HG11] H. Huang and G. Gartner. Using context-aware collaborative filtering for POI recommendations in mobile guides. In G. Gartner and F. Ortag, editors, *Eighth International Symposium on Location-Based Services (LBS): Advances in Location-Based Services*, Lecture Notes in Geoinformation and Cartography (LNGC), pp. 131–147. Springer, Berlin, Heidelberg, November 2011. ISBN 978-3-642-24198-7.

[HHB+03] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: Peer-to-peer media streaming using CollectCast. In *Eleventh ACM International Conference on Multimedia (MULTIMEDIA)*, pp. 45–54. ACM, New York (USA), November 2003. ISBN 1-58113-722-2.

[HIR02] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. In F. Mattern and M. Naghshineh, editors, *First International Conference on Pervasive Computing*, volume 2414 of *Lecture Notes in Computer Science (LNCS)*, pp. 167–180. Springer, Berlin, Heidelberg, August 2002. ISBN 978-3-540-45866-1.

[HITdCRH15] R. Hermoso, S. Ilarri, R. Trillo, and M. del Carmen Rodríguez-Hernández. Push-based recommendations in mobile computing using a multi-layer contextual approach. In *13th International Conference on Advances in Mo-*

bile Computing and Multimedia (MoMM), pp. 149–158. ACM, New York (USA), December 2015. ISBN 978-1-4503-3493-8.

[HK15] F. M. Harper and J. A. Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):1–19, December 2015. ISSN 2160-6455.

[HKBO16] P. Heshi, G. Kulkarni, K. Bidkar, and A. Oswal. Location aware recommendation system. *Imperial Journal of Interdisciplinary Research (IJIR)*, 2(6):1454–1457, 2016. ISSN 2454-1362.

[HKLK17] H. Hwangbo, J. Kim, Z. Lee, and S. Kim. Store layout optimization using indoor positioning system. *International Journal of Distributed Sensor Networks*, 13(2):1–13, February 2017.

[HKR02] J. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287–310, 2002. ISSN 1573-7659.

[HKTR04] 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, January 2004. ISSN 1046-8188.

[HLGZ14] T. Hussein, T. Linder, W. Gaulke, and J. Ziegler. Hybreed: A software framework for developing context-aware hybrid recommender systems. *User Modeling and User-Adapted Interaction*, 24(1):121–174, 2014. ISSN 1573-1391.

[HMB13] N. Hariri, B. Mobasher, and R. Burke. Query-driven context aware recommendation. In *Seventh ACM Conference on Recommender Systems (RecSys)*, pp. 9–16. ACM, New York (USA), 2013. ISBN 978-1-4503-2409-0.

[HNV06] T. Horozov, N. Narasimhan, and V. Vasudevan. Using location for personalized POI recommendations in mobile environments. In *International Symposium on Applications and the Internet (SAINT)*, pp. 124–129. IEEE, January 2006. ISBN 0-7695-2508-3.

[HNYL17] Z. Hao, E. Novak, S. Yi, and Q. Li. Challenges and software architecture for fog computing. *IEEE Internet Computing*, 21(2):44–53, March 2017. ISSN 1089-7801.

[HP02] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *28th International Conference on Very Large Data Bases (VLDB)*, pp. 670–681. VLDB Endowment, August 2002.

[HPC+16] T. Haute, E. Poorter, P. Crombez, F. Lemic, V. Handziski, et al. Performance analysis of multiple indoor positioning systems in a healthcare environment. *International Journal of Health Geographics*, 15(1):1–15, February 2016.

[HPNM09] M. Hosseini-Pozveh, M. Nematbakhsh, and N. Movahhedinia. A multidimensional approach for context-aware recommendation in mobile commerce. *CoRR*, abs/0908.0982, 2009.

[HR97] R. Hastings and M. Randolph. Netflix, 1997. American Entertainment Company, `https://www.netflix.com`, [Accessed on October 3, 2017].

[HR05] G. Hripcsak and A. S. Rothschild. Agreement, the F-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298, May 2005.

[HS09] C. Hoareau and I. Satoh. Modeling and processing information for context-aware computing: A survey. *New Generation Computing*, 27(3):177–196, May 2009. ISSN 1882-7055.

[HSG10] S. E. Helou, C. Salzmann, and D. Gillet. The 3A personalized, contextual and relation-based recommender system. *Journal of Universal Computer Science*, 16(16):2179–2195, August 2010.

[HSKK09] J. Hong, E.-H. Suh, J. Kim, and S. Kim. Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications*, 36(4):7448–7457, May 2009. ISSN 0957-4174.

[HSRF95] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *SIGCHI INternational Conference on Human Factors in Computing Systems (CHI)*, pp. 194–201. ACM, New York (USA), May 1995. ISBN 0-201-84705-1.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York (USA), second edition, 2001.

[HWBH16] P. Hiesel, W. Wörndl, M. Braunhofer, and D. Herzog. A user interface concept for context-aware recommender systems. *Mensch und Computer 2016-Tagungsband*, pp. 1–9, September 2016.

[HXYS04] P. Han, B. Xie, F. Yang, and R. Shen. A scalable P2P recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, 27(2):203–210, August 2004. ISSN 0957-4174.

[HY01] D. J. Hand and K. Yu. Idiot's Bayes – not so stupid after all? *International Statistical Review*, 69(3):385–398, December 2001.

[HZ11] N. Hurley and M. Zhang. Novelty and diversity in top-N recommendation – analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):1–30, March 2011. ISSN 1533-5399.

[IBHD16] M. D. Ingle, P. Bhor, G. Hargude, and S. Deshpande. Peer-to-peer video, audio streaming over Wi-Fi network on mobile device. *International Research Journal of Engineering and Technology (IRJET)*, 3(5):3279–3283, May 2016. ISSN 2395-0056.

[IBM15] IBM. IBM Quest Synthetic Data Generator, 2015. `http://ibmquestdatagen.sourceforge.net`, [Accessed on October 3, 2017].

[IDTL15] S. Ilarri, T. Delot, and R. Trillo-Lado. A data management perspective on vehicular networks. *IEEE Communications Surveys Tutorials*, 17(4):2420–2460, 2015. ISSN 1553-877X.

[IGL+08] L. Iaquinta, M. D. Gemmis, P. Lops, G. Semeraro, M. Filannino, et al. Introducing serendipity in a content-based recommender system. In *Eighth International Conference on Hybrid Intelligent Systems (HIS)*, pp. 168–173. IEEE, September 2008. ISBN 978-0-7695-3326-1.

[IHTLdCRH15] S. Ilarri, R. Hermoso, R. Trillo-Lado, and M. del Carmen Rodríguez-Hernández. A review of the role of sensors in mobile context-aware recommendation systems. *International Journal of Distributed Sensor Networks*, 11(11):1–30, January 2015. ISSN 1550-1329.

[IIMS11] S. Ilarri, A. Illarramendi, E. Mena, and A. Sheth. Semantics in location-based services – Guest editors' introduction for special issue. *IEEE Internet Computing*, 15(6):10–14, November 2011. ISSN 1089-7801.

[IKPC17] G. Ioannakis, A. Koutsoudis, I. Pratikakis, and C. Chamzas. RETRIEVAL–an online performance evaluation tool for information retrieval methods. *IEEE Transactions on Multimedia*, PP(99):1–9, 2017. ISSN 1941-0077.

[IMI10] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):12:1–12:73, March 2010. ISSN 0360-0300.

[IPCF17] N. Iotti, M. Picone, S. Cirani, and G. Ferrari. Improving quality of experience in future wireless access networks through fog computing. *IEEE Internet Computing*, 21(2):26–33, March 2017. ISSN 1089-7801.

[IWD14] S. Ilarri, O. Wolfson, and T. Delot. Collaborative sensing for urban transportation. *IEEE Data Engineering Bulletin*, 37(4):3–14, December 2014. Special Issue on Urban Informatics.

[JAK16] A. Jeyasekar, K. Akshay, and Karan. Collaborative filtering using euclidean distance in recommendation engine. *Indian Journal of Science and Technology*, 9(37):1–5, October 2016.

[Jam04] A. Jameson. More than the sum of its members: challenges for group recommender systems. In *International Working Conference on Advanced Visual Interfaces (AVI)*, pp. 48–54. ACM, New York (USA), May 2004. ISBN 1-58113-867-9.

[Jay90] A. Jay. IMDb, 1990. `http://www.imdb.com`, [Accessed on October 3, 2017].

[JDA+11] H. R. Jamali, Z. Dehghani, E. Afshar, H. R. Jamali, and M. A. Nematbakhsh. A multi-layer contextual model for recommender systems in digital libraries. *Aslib Journal of Information Management*, 63(6):555–569, 2011. ISSN 0001-253X.

[JDXB03] X. Jiang, Y. Dong, D. Xu, and B. Bhargava. GnuStream: a P2P media streaming system prototype. In *International Conference on Multimedia and Expo (ICME)*, volume 2, pp. 325–328. IEEE, July 2003. ISBN 0-7803-7965-9.

[JE10] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Fourth ACM Conference on Recommender Systems (RecSys)*, pp. 135–142. ACM, New York (USA), September 2010. ISBN 978-1-60558-906-0.

[Jes03] Jester. Jester Dataset, 2003. `http://goldberg.berkeley.edu/jester-data`, [Accessed on October 3, 2017].

[JGT+05] C. Jouvray, S. Gerard, F. Terrier, S. Bouaziz, and R. Reynaud. UML methodology for smart transducer integration in real-time embedded systems. In *IEEE Intelligent Vehicles Symposium*, pp. 688–693. IEEE, June 2005. ISBN 0-7803-8961-1. ISSN 1931-0587.

[jHRJH10] B. jun Han, S. Rho, S. Jun, and E. Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, May 2010. ISSN 1573-7721.

[JK02] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, October 2002. ISSN 1046-8188.

[jKAJ10] K. jae Kim, H. Ahn, and S. Jeong. Context-aware recommender systems using data mining techniques. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 4(4):381–3862, 2010. ISSN 2010-3778.

[JKG12] D. Jannach, Z. Karakaya, and F. Gedikli. Accuracy improvements for multi-criteria recommender systems. In *13th ACM Conference on Electronic Commerce (EC*, pp. 674–689. ACM, New York (USA), June 2012. ISBN 978-1-4503-1415-2.

[JKSS07] S. Jbara, T. Kuflik, P. Soffer, and O. Stock. Context aware communication services in "active museums". In *IEEE International Conference on Software-Science, Technology & Engineering (SwSTE)*, pp. 127–135. IEEE, October 2007. ISBN 0-7695-3021-4.

[JL95] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *11th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 338–345. Morgan Kaufmann, San Francisco (USA), August 1995. ISBN 1-55860-385-9.

[JMN+16] K. Jacobson, V. Murali, E. Newett, B. Whitman, and R. Yon. Music personalization at Spotify. In *10th ACM Conference on Recommender Systems (RecSys)*, pp. 373–373. ACM, New York (USA), September 2016. ISBN 978-1-4503-4035-9.

[JS07] A. Jameson and B. Smyth. Recommendation to groups. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science (LNCS)*, chapter 20, pp. 596–627. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9.

[KABO10] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Fourth ACM Conference on Recommender Systems (RecSys)*, pp. 79–86. ACM, New York (USA), September 2010. ISBN 978-1-60558-906-0.

[Kak04] Kakaku.com Company. Tabelog, 2004. `https://tabelog.com/`, [Accessed on October 3, 2017].

[KBCB12] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer. Climbing the app wall: Enabling mobile app discovery through context-aware recommendations. In *21st ACM International Conference on Information and*

Knowledge Management (CIKM), pp. 2527–2530. ACM, New York (USA), October 2012. ISBN 978-1-4503-1156-4.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009. ISSN 0018-9162.

[KCL09] M.-H. Kuo, L.-C. Chen, and C.-W. Liang. Building and evaluating a location-based service recommendation system with a preference adjustment mechanism. *Expert Systems with Applications*, 36(2):3543–3554, March 2009. ISSN 0957-4174.

[KDK11] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 165–172. ACM, New Yor (USA), October 2011. ISBN 978-1-4503-0683-6.

[Kee10] B. Keen. Generatedata.com, 2010. `http://www.generatedata.com`, `https://github.com/benkeen/generatedata`, [Accessed on October 3, 2017].

[Ken38] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, June 1938.

[KG17] J. Kargar and A. Ghasemi. A new mechanism to improve video streaming in P2P networks using helper nodes. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25(2):877–887, 2017.

[KGW00] J. Kraft, W. Glaser, and T. Westergren. Pandora, 2000. Pandora Internet Radio and Music Genome Project, `http://www.pandora.com`, [Accessed on October 3, 2017].

[KIRD17] R. Karchoud, S. Ilarri, P. Roose, and M. Dalmau. Long life application: Situation detection in a context-aware all-in-one application. *Personal and Ubiquitous Computing*, 2017. ISSN 1617-4917. To appear.

[KKC08] J. K. Kim, H. K. Kim, and Y. H. Cho. A user-oriented contents recommendation system in peer-to-peer architecture. *Expert Systems with Applications*, 34(1):300–312, January 2008. ISSN 0957-4174.

[KLK16] A. Kim, J. Lee, and M. Kim. Context-aware recommendation model based on mobile application analysis platform. *Multimedia Tools and Applications*, 75(22):14783–14794, 2016. ISSN 1573-7721.

[KLP+06] B. M. Kim, Q. Li, C. S. Park, S. G. Kim, and J. Y. Kim. A new approach for combining content-based and collaborative filters. *Journal of Intelligent Information Systems*, 27(1):79–91, August 2006. ISSN 1573-7675.

[KLZ12] S. M. Kywe, E.-P. Lim, and F. Zhu. A survey of recommender systems in Twitter. In K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, et al., editors, *Fourth International Conference on Social Informatics (SocInfo)*, volume 7710 of *Lecture Notes in Computer Science (LNCS)*, pp. 420–433. Springer, Berlin, Heidelberg, December 2012. ISBN 978-3-642-35386-4.

[KM99] A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. In M. Mohammadian, editor, *International Conference on Computational Intelligence for Modeling, Control & Automation (CIMCA)*, pp. 199–203. IOS Press, Amsterdam (Netherlands), February 1999. ISBN 90-5199-475-3.

[KM16] E. Karydi and K. Margaritis. Parallel and distributed collaborative filtering: A survey. *ACM Computing Surveys (CSUR)*, 49(2):1–37, August 2016. ISSN 0360-0300.

[KMM+97] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, et al. GroupLens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, March 1997. ISSN 0001-0782.

[KMtH06] R. Kramer, M. Modsching, and K. ten Hagen. Field study on methods for elicitation of preferences using a mobile digital assistant for a dynamic tour guide. In *2006 ACM Symposium on Applied Computing (SAC)*, pp. 997–1001. ACM, New York (USA), April 2006. ISBN 1-59593-108-2.

[KN10] G. Kreitz and F. Niemela. Spotify – large scale, low latency, P2P music-on-demand streaming. In *IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–10. IEEE, August 2010. ISBN 978-1-4244-7141-6. ISSN 2161-3559.

[Koh95] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *14th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pp. 1137–1143. Stanford, CA, August 1995.

[KOK+11] A. Košir, A. Odić, M. Kunaver, M. Tkalčič, and J. F. Tasič. Database for contextual personalization. *Elektrotehniški vestnik*, 78(5):270–274, 2011.

[Košl1] A. Košir. LDOS-CoMoDa dataset, 2011. `http://www.ldos.si/comoda.html`, [Accessed on October 3, 2017].

[KP10] C. Kaleli and H. Polat. P2P collaborative filtering with privacy. *Turkish Journal of Electrical Engineering & Computer Sciences*, 18(1):101–116, 2010.

[KPJ06] S. Kabadayi, A. Pridgen, and C. Julien. Virtual sensors: Abstracting data from physical sensors. In *International Symposium on World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, pp. 587–592. IEEE, Washington (USA), June 2006. ISBN 0-7695-2593-8.

[KRR17] S. Keshvadi, A. M. Rahmani, and H. Rostami. Recommend top-K most downloaded files in the chord-based P2P file-sharing system. *Peer-to-Peer Networking and Applications*, 10(1):208–215, January 2017. ISSN 1936-6450.

[KRS13a] M. Kaminskas, F. Ricci, and M. Schedl. Location-aware music recommendation using auto-tagging and hybrid matching. In *Seventh ACM Conference on Recommender Systems (RecSys)*, pp. 17–24. ACM, New York (USA), October 2013. ISBN 978-1-4503-2409-0.

[KRS13b] H.-N. Kim, M. Rawashdeh, and A. E. Saddik. Tailoring recommendations to groups of users: a graph walk-based approach. In *International Conference on Intelligent User Interfaces (IUI)*, pp. 15–24. ACM, New York (USA), March 2013. ISBN 978-1-4503-1965-2.

[KS+00] S. Kaufer, L. Steinert, et al. TripAdvisor, 2000. `https://www.tripadvisor.com`, [Accessed on October 3, 2017].

[KS12] L. Kazemi and C. Shahabi. GeoCrowd: Enabling query answering with spatial crowdsourcing. In *20th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pp. 189–198. ACM, New York (USA), November 2012. ISBN 978-1-4503-1691-0.

[KS16] B. Kumar and N. Sharma. Approaches, issues and challenges in recommender systems: A systematic review. *Indian Journal of Science and Technology*, 9(47):1–12, December 2016. ISSN 0974-5645.

[KSP+01] G. Kortuem, J. Schneider, D. Preuitt, T. G. C. Thompson, S. Fickas, et al. When peer-to-peer comes Face-to-Face: Collaborative peer-to-peer computing in mobile ad-hoc networks. In *First International Conference on Peer-to-Peer Computing*, pp. 75–91. IEEE, August 2001. ISBN 0-7695-1503-7.

[KvWG09] M. Kagie, M. van Wezel, and P. J. F. Groenen. An empirical comparison of dissimilarity measures for recommender systems. ERIM Report Series ERS-2009-023-MKT, Erasmus Research Institute of Management, Erasmus School of Economics, May 2009.

[KWPG01] S. Kramer, G. Widmer, B. Pfahringer, and M. D. Groeve. Prediction of ordinal classes using regression trees. *Foundations of Intelligent Systems*, 47(1-2):1–13, September 2001.

[KWV16] D. Kotkov, S. Wang, and J. Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192, 2016. ISSN 0950-7051.

[KZP07] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas. Supervised Machine Learning: A review of classification techniques. In I. Maglogiannis, K. Karpouzis, M. Wallace, and J. Soldatos, editors, *Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, volume 160, pp. 249–268. IOS Press, 2007. ISBN 978-1-58603-780-2.

[LA13] X. Liu and K. Aberer. SoCo: A social network aided context-aware recommender system. In *22nd International Conference on World Wide Web (WWW)*, pp. 781–802. ACM, New York (USA), May 2013. ISBN 978-1-4503-2035-1.

[Lat15] N. Lathia. The anatomy of mobile location-based recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, chapter 14, pp. 493–510. Springer, Boston, MA, second edition, 2015. ISBN 978-1-4899-7637-6.

[LC08] G. Lekakos and P. Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36(1):55–70, January 2008. ISSN 1573-7721.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th International Conference on Supercomputing (ICS)*, pp. 84–95. ACM, New York (USA), June 2002. ISBN 1-58113-483-5.

[LCCT17] J. Li, C. Chen, H. Chen, and C. Tong. Towards context-aware social recommendation via individual trust. *Knowledge-Based Systems*, 127:58–66, July 2017. ISSN 0950-7051.

[LCF+17] D. Lanza, F. Chávez, F. Fernandez, M. Garcia-Valdez, L. Trujillo, et al. Profiting from several recommendation algorithms using a scalable approach. In O. Schütze, L. Trujillo, P. Legrand, and Y. Maldonado, editors, *Results of the Numerical and Evolutionary Optimization Workshop NEO*, volume 663 of *Studies in Computational Intelligence (SCI)*, pp. 357–375. Springer, Cham, September 2017. ISBN 978-3-319-44003-3.

[LClC06] C. W. K. Leung, S. C. F. Chan, and F. lai Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In A. Felfernig and M. Zanker, editors, *International Workshop on Recommender Systems, in conjunction with the 17th European Conference on Artificial Intelligence (ECAI)*, pp. 62–66. August 2006.

[LCLS10] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th International Conference on World Wide Web (WWW)*, pp. 661–670. ACM, New York (USA), April 2010. ISBN 978-1-60558-799-8.

[LCLW11] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 297–306. ACM, New York (USA), February 2011. ISBN 978-1-4503-0493-1.

[LCVA12] F. D. A. Lemos, R. A. F. Carmo, W. Viana, and R. M. C. Andrade. Towards a context-aware photo recommender system. In *Fourth Workshop on Context-Aware Recommender Systems (CARS)*, pp. 1–5. CEUR Workshop Proceedings, September 2012. ISBN 978-1-4503-1270-7. ISSN 1613-0073.

[LD06] A. Loizou and S. Dasmahapatra. Recommender systems for the Semantic Web. In *ECAI 2006 Recommender Systems Workshop*, pp. 76–81. August 2006.

[LGL08] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28, March 2008. ISSN 1936-6450.

[LGS11a] P. Lops, M. D. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 3, pp. 73–105. Springer, Boston, MA, 2011. ISBN 978-0-387-85820-3.

[LGS+11b] P. Lops, M. D. Gemmis, G. Semeraro, F. Narducci, and C. Musto. Leveraging the LinkedIn social network data for extracting content-based user profiles. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 293–296. ACM, New York (USA), October 2011. ISBN 978-1-4503-0683-6.

[Li04] J. Li. PeerStreaming: A practical receiver-driven peer-to-peer media streaming system. Microsoft Research TechReport MSR-TR-2004-101, One Microsoft Way, September 2004.

[Lin14] Z. Lin. An empirical investigation of user and system recommendations in e-commerce. *Decision Support Systems*, 68:111–124, 2014. ISSN 0167-9236.

[LIT92] P. Langley, W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *10th National Conference on Artificial Intelligence (AAAI)*, volume 90, pp. 223–228. AAAI Press, July 1992. ISBN 0-262-51063-4.

[Liu10] B. Liu. Sentiment analysis and subjectivity. In N. Indurkhya and F. J. Damerau, editors, *Handbook of Natural Language Processing*, Machine Learning & Pattern Recognition, chapter 26, pp. 627–666. Chapman and Hall/CRC, Boca Raton, Florida (USA), second edition, February 2010. ISBN 978-1-4200-8592-1.

[Liu11] B. Liu. *Opinion Mining and Sentiment Analysis*, chapter 11, pp. 459–526. Data-Centric Systems and Applications (DCSA). Springer, Berlin, Heidelberg, second edition, April 2011. ISBN 978-3-642-19459-7.

[Liu14] Q. Liu. Context-aware mobile recommendation system based on context history. *Indonesian Journal of Electrical Engineering and Computer Science*, 12(4):3158–3167, April 2014. ISSN 2502-4760.

[LJdVT11] Y. Lin, J. Jessurun, B. de Vries, and H. Timmermans. Motivate: Towards context-aware recommendation mobile system for healthy living. In *Fifth International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pp. 250–253. IEEE, May 2011. ISBN 978-1-936968-15-2. ISSN 2153-1633.

[LK14] H. Lee and J. Kwon. Personalised TV contents recommender system using collaborative context tagging-based users's preference prediction technique. *International Journal of Multimedia & Ubiquitous Engineering*, 9(5):231–240, 2014.

[LKH14a] W.-P. Lee, C. Kaoli, and J.-Y. Huang. A smart TV system with body-gesture control, tag-based rating and context-aware recommendation. *Knowledge-Based Systems*, 56:167–178, January 2014. ISSN 0950-7051.

[LKH14b] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014. ISSN 0957-4174.

[LL06] J. Lee and J. Lee. Music for my mood: A music recommendation system based on context reasoning. In P. Havinga, M. Lijding, N. Meratnia, and M. Wegdam, editors, *First European Conference on Smart Sensing and Context (EuroSSC)*, volume 4272 of *Lecture Notes in Computer Science (LNCS)*, pp. 190–203. Springer, Berlin, Heidelberg, October 2006. ISBN 978-3-540-47845-4.

[LL07] J. Lee and J. Lee. Context awareness by case-based reasoning in a music recommendation system. In H. Ichikawa, W.-D. Cho, I. Satoh, and H. Y. Youn, editors, *Fourth International Symposium on Ubiquitous Computing Systems (UCS)*, volume 4836 of *Lecture Notes in Computer Science (LNCS)*, pp. 45–58. Springer, Berlin, Heidelberg, November 2007. ISBN 978-3-540-76772-5.

[LLH17] Y.-M. Li, L.-F. Lin, and C.-C. Ho. A social route recommender mechanism for store shopping support. *Decision Support Systems*, 94:97–108, 2017. ISSN 0167-9236.

[LM05] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SIAM International Conference on Data Mining*, pp. 471–475. SIAM, March 2005. ISBN 978-1-61197-275-7.

[LMA+16] G. Larkou, M. Mintzis, P. G. Andreou, A. Konstantinidis, and D. Zeinalipour-Yazti. Managing Big Data experiments on smartphones. *Distributed and Parallel Databases*, 34(1):33–64, 2016. ISSN 1573-7578.

[LMCX13] Q. Liu, H. Ma, E. Chen, and H. Xiong. A survey of context-aware mobile recommendations. *International Journal of Information Technology & Decision Making*, 12(1):139–172, 2013.

[LML+10] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, et al. A survey of mobile phone sensing. *Communications Magazine*, 48(9):140–150, September 2010. ISSN 0163-6804.

[LMMSC14] I. M. Lombera, L. E. Moser, P. M. Melliar-Smith, and Y.-T. Chuang. Peer-to-peer publication, search and retrieval using the Android mobile platform. *Computer Networks*, 65:56–72, February 2014. ISSN 1389-1286.

[LMV00] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[LMY+12] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, et al. Recommender systems. *Physics Reports*, 519(1):1–49, 2012. ISSN 0370-1573.

[LMZW10] X. Li, Z. Mi, Z. Zhang, and J. Wu. A location-aware recommender system for tourism mobile commerce. In *Second International Conference on Information Science and Engineering (ICISE)*, pp. 1709–1711. IEEE, December 2010. ISBN 978-1-4244-7618-3. ISSN 2160-1283.

[LOF+08] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 903–914. ACM, New York (USA), June 2008. ISBN 978-1-60558-102-6.

[Lou95] H.-L. Lou. Implementing the Viterbi algorithm. *Signal Processing Magazine, IEEE*, 12(5):42–52, September 1995. ISSN 1053-5888.

[LP07] H. Lee and S. J. Park. MONERS: A news recommender for the mobile web. *Expert Systems with Applications*, 32(1):143 – 150, January 2007. ISSN 0957-4174.

[LPCY17] Y. Liu, T.-A. N. Pham, G. Cong, and Q. Yuan. An experimental evaluation of Point-of-Interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment*, 10(10):1010–1021, June 2017. ISSN 2150-8097.

[LPP10] T. Lu, D. Pal, and M. Pal. Contextual multi-armed bandits. In Y. W. Teh and M. Titterington, editors, *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *Proceedings of Machine*

Learning Research, pp. 485–492. PMLR, Chia Laguna Resort, Sardinia (Italy), May 2010.

[LRHW15] B. Lamche, Y. Rödl, C. Hauptmann, and W. Wörndl. Context-aware recommendations for mobile shopping. In *International Workshop on Location-Aware Recommendations (LocalRec) co-located, in conjunction with the Ninth ACM Conference on Recommender Systems (RecSys)*, volume 1405, pp. 21–27. CEUR Workshop Proceedings, September 2015. ISSN 1613-0073.

[LRU14] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Recommendation Systems*, chapter 9, pp. 292–324. Cambridge University Press, second edition, 2014. ISBN 978-1-107-07723-2.

[LSEM12] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. LARS: A location-aware recommender system. In *28th International Conference on Data Engineering (ICDE)*, pp. 450–461. IEEE, April 2012. ISBN 978-1-4673-0042-1. ISSN 1063-6382.

[LSS07] J.-S. Lee, Y.-W. Su, and C.-C. Shen. A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In *33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, pp. 46–51. IEEE, November 2007. ISBN 1-4244-0783-4. ISSN 1553-572X.

[LST12] A. S. Lampropoulos, D. N. Sotiropoulos, and G. A. Tsihrintzis. Evaluation of a cascade hybrid recommendation as a combination of one-class classification and collaborative filtering. In *24th International Conference on Tools with Artificial Intelligence*, volume 1, pp. 674–681. IEEE, November 2012. ISSN 1082-3409.

[LST14] A. S. Lampropoulos, D. N. Sotiropoulos, and G. A. Tsihrintzis. Cascade hybrid recommendation as a combination of one-class classification and collaborative filtering. *International Journal on Artificial Intelligence Tools*, 23(4):1460009, August 2014.

[LSY03] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, January 2003. ISSN 1089-7801.

[LT07] H.-H. Lee and W.-G. Teng. Incorporating multi-criteria ratings in recommendation systems. In *2007 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 273–278. IEEE, August 2007. ISBN 1-4244-1499-7.

[LT16] W.-P. Lee and G.-Y. Tseng. Incorporating contextual information and collaborative filtering methods for multimedia recommendation in a mobile environment. *Multimedia Tools and Applications*, 75(24):16719–16739, 2016. ISSN 1573-7721.

[LVLD08] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. In *Second International Conference on Ubiquitous Information Management and Communication (ICUIMC)*, pp. 208–211. ACM, New York (USA), January 2008. ISBN 978-1-59593-993-7.

[LW08] Y. Luo and O. Wolfson. Mobile P2P databases. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, pp. 671–677. Springer, Boston, MA, 2008. ISBN 978-0-387-35973-1.

[LW12] B. Lerchenmueller and W. Woerndl. Inference of user context from GPS logs for proactive recommender systems. AAAI Technical Report WS-12-05, Technische Universitaet Muenchen, July 2012.

[LWGD07] Q. Li, C. Wang, G. Geng, and R. Dai. A novel collaborative filtering-based framework for personalized services in m-commerce. In *16th International Conference on World Wide Web (WWW)*, pp. 1251–1252. ACM, New York (USA), May 2007. ISBN 978-1-59593-654-7.

[LWM+15] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang. Recommender system application developments: A survey. *Decision Support Systems*, 74(Supplement C):12–32, 2015. ISSN 0167-9236.

[LZ12] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, chapter 13, pp. 415–463. Springer, Boston, MA, January 2012. ISBN 978-1-4614-3223-4.

[MA07] P. Massa and P. Avesani. Trust-aware recommender systems. In *ACM Conference on Recommender Systems (RecSys)*, pp. 17–24. ACM, New York (USA), October 2007. ISBN 978-1-59593-730–8.

[MA15] P. Moradi and S. Ahmadian. A reliability-based recommendation method to improve trust-aware recommender systems. *Expert Systems with Applications*, 42(21):7386–7398, 2015. ISSN 0957-4174.

[Maj12] P. Maj. DBMonster, 2012. `http://dbmonster.sourceforge.net`, [Accessed on October 3, 2017].

[MAP16] N. Mendes, B. Alves, and S. Paiva. Context-aware mobile recommender system based on activity patterns. In *17th IEEE International Conference on Mobile Data Management (MDM)*, volume 2, pp. 70–74. IEEE, June 2016. ISBN 978-1-5090-0883-4. ISSN 2375-0324.

[MAV17] M. B. Mollah, M. A. K. Azad, and A. Vasilakos. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, pp. –, 2017. ISSN 1084-8045.

[MBBW07] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology*, 7(4):23:1–23:41, October 2007. ISSN 1533-5399.

[MBDR10] M. Mordacchini, R. Baraglia, P. Dazzi, and L. Ricci. A P2P recommender system based on gossip overlays (PREGO). In *10th IEEE International Conference on Computer and Information Technology*, pp. 83–90. IEEE, June 2010. ISBN 978-1-4244-7548-3.

[MBR09] M. C. P. Melguizo, L. Boves, and O. M. Ramos. A proactive recommendation system for writing: Helping without disrupting. *International Journal of Industrial Ergonomics*, 39(3):516–523, 2009. ISSN 0169-8141.

[MCG+99] T. Miranda, M. Claypool, A. Gokhale, T. Mir, P. Murnikov, et al. Combining content-based and collaborative filters in an online newspaper. Computer Science Technical Report WPI-CS-TR-99-16, Worcester Polytechnic Institute, June 1999.

[MCR11] A. Moreno, H. Castro, and M. Riveill. Decentralized recommender systems for mobile advertisement. In *International Workshop on Personalization in Mobile Applications (PeMA), in conjunction with the ACM International Conference on Recommender Systems (RecSys)*, pp. 1–4. October 2011.

[MDA05] G. McLachlan, K.-A. Do, and C. Ambroise. *Analyzing Microarray Gene Expression Data*, volume 422. John Wiley & Sons, 2005. ISBN 0-471-22616-5.

[MH04] M. R. McLaughlin and J. L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 329–336. ACM, New York (USA), July 2004. ISBN 1-58113-881-4.

[MHF07] B. Mehta, T. Hofmann, and P. Fankhauser. Lies and propaganda: Detecting spam users in collaborative filtering. In *12th International Conference on Intelligent User Interfaces (IUI)*, pp. 14–21. ACM, New York (USA), January 2007. ISBN 1-59593-481-2.

[Mil04] J. Miller. Characterization of data on the Gnutella peer-to-peer network. In *First IEEE Consumer Communications and Networking Conference*, pp. 489–494. IEEE, January 2004. ISBN 0-7803-8145-9.

[MKL09] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 203–210. ACM, New York (USA), July 2009. ISBN 978-1-60558-483-6.

[MKS11] A. Machanavajjhala, A. Korolova, and A. D. Sarma. Personalized social recommendations: Accurate or private. *Proceedings of the VLDB Endowment*, 4(7):440–450, April 2011. ISSN 2150-8097.

[ML10] B. McFee and G. R. Lanckriet. Metric learning to rank. In *27th International Conference on Machine Learning (ICML)*, pp. 775–782. June 2010.

[MLCM13] K. Meehan, T. Lunney, K. Curran, and A. McCaughey. Context-aware intelligent recommendation system for tourism. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 328–331. IEEE, March 2013. ISBN 978-1-4673-5077-8.

[MLF+08] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, et al. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Sixth ACM Conference on Embedded Network Sensor Systems (SenSys)*, pp. 337–350. ACM, New York (USA), November 2008. ISBN 978-1-59593-990-6.

[MLM+17] X. Ma, H. Li, J. Ma, Q. Jiang, S. Gao, et al. APPLET: a privacy-preserving framework for location-aware recommender system. *Science China Information Sciences*, 60(9):1–16, October 2017. ISSN 1869-1919.

[MN13] G. W. Musumba and H. O. Nyongesa. Context awareness in mobile computing: A review. *International Journal of Machine Learning and Applications*, 2(1):1–10, May 2013.

[MP00] K. Miyahara and M. Pazzani. Collaborative filtering with the simple Bayesian classifier. In R. Mizoguchi and J. Slaney, editors, *Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI): Topics in Artificial Intelligence*, volume 1886 of *Lecture Notes in Computer Science (LNCS)*, pp. 679–689. Springer, Berlin, Heidelberg, August 2000. ISBN 978-3-540-44533-3.

[MP13] C. Mettouris and G. Papadopoulos. Contextual modelling in context-aware recommender systems: A generic approach. In A. Haller, G. Huang, Z. Huang, H. young Paik, and Q. Z. Sheng, editors, *2011 and 2012 Web Information Systems Engineering (WISE) Workshops*, volume 7652 of *Lecture Notes in Computer Science (LNCS)*, pp. 41–52. Springer, Berlin, Heidelberg, 2013. ISBN 978-3-642-38333-5.

[MP14] C. Mettouris and G. Papadopoulos. Ubiquitous recommender systems. *Computing*, 96(3):223–257, 2014. ISSN 1436-5057.

[MPCB16] A. Morris, C. Patsakis, V. Cahill, and M. Bouroche. Snapcab: Urban scale context-aware smart transport using adaptive context tries. In P. C. Vinh and V. Alagar, editors, *Fourth International Conference on Context-Aware Systems and Applications (ICCASA)*, volume 165 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST)*, pp. 31–40. Springer, Cham, November 2016. ISBN 978-3-319-29236-6.

[MPR+12] U. Morbiducci, R. Ponzini, G. Rizzo, M. E. Biancolini, F. Iannaccone, et al. Synthetic dataset generation for the analysis and the evaluation of image-based hemodynamics of the human aorta. *Medical & Biological Engineering & Computing*, 50(2):145–154, 2012. ISSN 1741-0444.

[MPS14] L. Mainetti, L. Patrono, and I. Sergi. A survey on indoor positioning systems. In *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pp. 111–120. IEEE, September 2014. ISBN 978-9-5329-0052-1.

[MR00] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Fifth ACM Conference on Digital Libraries (DL)*, pp. 195–204. ACM, New York (USA), June 2000. ISBN 1-58113-231-X.

[MRK06] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *International Conference on Human Factors in Computing Systems and Extended Abstracts (CHI EA)*, pp. 1097–1101. ACM, New York (USA), April 2006. ISBN 1-59593-298-4.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*, volume 1. Cambridge University Press Cambridge, 2008. ISBN 978-0-521-86571-5.

[MRS+09] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, et al. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In *Seventh International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 55–68. ACM, ACM, New York (USA), June 2009. ISBN 978-1-60558-566-6.

[MSLdG14] C. Musto, G. Semeraro, P. Lops, and M. de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, et al., editors, *22nd International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, volume 8538 of *Lecture Notes in Computer Science (LNCS)*, pp. 381–392. Springer, Cham, July 2014. ISBN 978-3-319-08786-3.

[MTAS17] V. Maccatrozzo, M. Terstall, L. Aroyo, and G. Schreiber. SIRUP: Serendipity in recommendations via user perceptions. In *22nd International Conference on Intelligent User Interfaces (IUI)*, pp. 35–44. ACM, New York (USA), March 2017. ISBN 978-1-4503-4348-0.

[MXHA05] M. F. Mokbel, X. Xiong, M. A. Hammad, and W. G. Aref. Continuous query processing of spatio–temporal data streams in PLACE. *GeoInformatica*, 9(4):343–365, December 2005. ISSN 1573-7624.

[MYDÁlG+16] C. Marcelo, C. Yot-Domínguez, J. A. Á lvarez García, J. A. Ortega-Ramírez, and Á. l. Arcos-García. Learning in mobility with Context4Learning: developing a context-aware mobile learning application. *International Journal of Mobile Learning and Organisation*, 10(4):203–222, 2016. ISSN 1746-7268.

[MYLK08] H. Ma, H. Yang, M. R. Lyu, and I. King. SoRec: Social recommendation using probabilistic matrix factorization. In *17th ACM Conference on Information and Knowledge Management (CIKM)*, pp. 931–940. ACM, New York (USA), October 2008. ISBN 978-1-59593-991-3.

[MYZ+17] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. Mobile edge computing: Survey and research outlook. *ArXiv e-prints*, abs/1701.01090, January 2017.

[MZ09] B. M. Marlin and R. S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 5–12. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[MZL+11] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 287–296. ACM, New York (USA), February 2011. ISBN 978-1-4503-0493-1.

[NACH17] G. Nicolas, T. Amghar, O. Camp, and S. Hammoudi. A framework for context-aware service recommendation for mobile users: A focus on mobility in smart cities. *From Data To Decision*, pp. 1–17, 2017.

[NBM+06] T. Neumann, M. Bender, S. Michel, G. Weikum, P. Bonnet, et al. A reproducible benchmark for P2P retrieval. In *First International Workshop on Performance and Evaluation of Data Management Systems (ExpDB)*, pp. 1–8. ACM, 2006. Http://hdl.handle.net/11858/00-001M-0000-000F-221F-7.

[NBSM12] J. M. Noguera, M. J. Barranco, R. Segura, and L. Martínez. A location-aware tourism recommender system based on mobile devices. In C. Kahraman, E. E. Kerre, and F. T. Bozbura, editors, *10th International FLINS*

Conference: Uncertainty Modeling in Knowledge Engineering and Decision Making, volume 7 of *Computer Engineering and Information Science*, pp. 34–39. World Scientific, August 2012. ISBN 978-981-4417-73-0.

[NCI+11] A. Nika, T. Catarci, Y. Ioannidis, A. Katifori, G. Koutrika, et al. A survey of context-aware cross-digital library personalization. In M. Detyniecki, P. Knees, A. Nürnberger, M. Schedl, and S. Stober, editors, *Eighth International Workshop on Adaptive Multimedia Retrieval (AMR)*, volume 6817 of *Lecture Notes in Computer Science (LNCS)*, pp. 16–30. Springer, Berlin, Heidelberg, August 2011. ISBN 978-3-642-27169-4.

[Net09] Netflix. Netflix Prize Dataset, 2009. `http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a`, [Accessed on October 3, 2017].

[NGL11] J. Naruchitparames, M. H. Güneş, and S. J. Louis. Friend recommendations in social networks using genetic algorithms and network topology. In *IEEE Congress on Evolutionary Computation (CEC)*, pp. 2207–2214. IEEE, June 2011. ISBN 978-1-4244-7835-4. ISSN 1089-778X.

[NIIZ15] M. Nilashi, O. B. Ibrahim, N. Ithnin, and R. Zakaria. A multi-criteria recommendation system using dimensionality reduction and neuro-fuzzy techniques. *Soft Computing*, 19(11):3173–3207, November 2015. ISSN 1433-7479.

[NKG15] A. N. Nikolakopoulos, M. A. Kouneli, and J. D. Garofalakis. Hierarchical itemspace rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation. *Neurocomputing*, 163:126–136, 2015. ISSN 0925-2312.

[NMJ+13] J. K. Nurminen, A. J. R. Meyn, E. Jalonen, Y. Raivio, and R. G. Marrero. P2P media streaming with HTML5 and WebRTC. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 63–64. IEEE, April 2013. ISBN 978-1-4799-0056-5.

[NTCS99] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom): Wireless Networks*, volume 8, pp. 151–162. ACM, Secaucus (USA), March 1999. ISBN 1-58113-142-9. ISSN 1022-0038.

[NWS17] M. R. Nowicki, J. Wietrzykowski, and P. Skrzypczyński. Real-time visual place recognition for personal localization on a mobile device. *Wireless Personal Communications*, pp. 1–32, May 2017. ISSN 1572-834X.

[OCKR01] M. O'connor, D. Cosley, J. A. Konstan, and J. Riedl. PolyLens: A recommender system for groups of users. In W. Prinz, M. Jarke, Y. Rogers, K. Schmidt, and V. Wulf, editors, *Seventh European Conference on Computer Supported Cooperative Work (ECSCW)*, pp. 199–218. Springer, Dordrecht, September 2001. ISBN 978-0-306-48019-5.

[Omi95] P. Omidyar. eBay, 1995. Multinational E-commerce Corporation, `http://www.ebay.com`, [Accessed on October 3, 2017].

[ONM+12] V. C. Ostuni, T. D. Noia, R. Mirizzi, D. Romito, and E. D. Sciascio. Cinemappy: A context-aware mobile app for movie recommendations boosted by DBpedia. In *International Workshop on Semantic Technologies Meet*

Recommender Systems & Big Data (SeRSy), in conjunction with the 11th International Semantic Web Conference (ISWC), volume 919, pp. 37–48. CEUR Workshop Proceedings, Aachen (Germany), November 2012. ISSN 1613-0073.

[ONMU06] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware SVM for context-dependent information recommendation. In *Seventh International Conference on Mobile Data Management (MDM)*, pp. 109–109. IEEE, Washington (USA), May 2006. ISBN 0-7695-2526-1.

[ONMU07] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Investigation for designing of context-aware recommendation system using SVM. In *International MultiConference of Engineers and Computer Scientists (IMECS)*, pp. 970–975. March 2007.

[Ope04] OpenStreetMap. Nominatim, 2004. `http://nominatim.openstreetmap.org`, [Accessed on October 3, 2017].

[Ora01] A. Oram. *Peer-to-Peer: Harnessing the power of disruptive technologies*. O'Reilly Media, Inc., first edition, 2001. ISBN 0-596-00110-X.

[OTMA09] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. In G.-J. Houben, G. McCalla, F. Pianesi, and M. Zancanaro, editors, *17th International Conference on User Modeling, Adaptation and Personalization (UMAP)*, volume 5535 of *Lecture Notes in Computer Science (LNCS)*, pp. 102–113. Springer, June 2009. ISBN 978-3-642-02246-3.

[OTTK11] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. In *Fourth Workshop on Context-Aware Recommender Systems (CARS)*, volume 889. CEUR Workshop Proceedings, September 2011. ISSN 1613-0073.

[OTTK13] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[Pas05] A. Pashtan. *Mobile Web Services*. Cambridge University Press, 2005. ISBN 0-521-83049-4.

[Paz99] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, December 1999. ISSN 1573-7462.

[PB07] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science (LNCS)*, chapter 10, pp. 325–341. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9.

[PBV05] B. Pourebrahimi, K. Bertels, and S. Vassiliadis. A survey of peer-to-peer networks. In *16th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC)*, volume 2005. Citeseer, November 2005. ISBN 907-3-46150-2.

[PC09] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 21–28. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[PCV+16] T. D. Pessemier, C. Courtois, K. Vanhecke, K. V. Damme, L. Martens, et al. A user-centric evaluation of context-aware recommendations for a mobile news service. *Multimedia Tools and Applications*, 75(6):3323–3351, 2016. ISSN 1573-7721.

[PDM17] T. D. Pessemier, J. Dhondt, and L. Martens. Hybrid group recommendations for a travel service. *Multimedia Tools and Applications*, 76(2):2787–2811, 2017. ISSN 1573-7721.

[PdSS16] J. Pes, J. da Silva, and E. Sharpe. Visitor figures 2015 – the grand totals: Exhibition and museum attendance numbers worldwide. *The Art Newspaper*, XXV(278), April 2016. Special report. http://www.museus.gov.br/wp-content/uploads/2016/04/Visitor-Figures-2015-LO.pdf.

[PEAZ13] M. Pasinato, C. M. Eduardo, M.-A. Aufaure, and G. Zimbrão. Generating synthetic data for context-aware recommender systems. In *BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC)*, pp. 563–567. IEEE, September 2013. ISBN 978-1-4799-3194-1. ISSN 2377-0589.

[PF99] S. Parker and S. Fanning. Napster, 1999. `http://www.napster.com`, [Accessed on October 3, 2017].

[PFS17] J. Poderys, J. Farooq, and J. Soler. A multimedia streaming system for urban rail environments. In A. Pirovano, M. Berbineau, A. Vinel, C. Guerber, D. Roque, et al., editors, *International Workshop on Communication Technologies for Vehicles*, volume 10222 of *Lecture Notes in Computer Science (LNCS)*, pp. 41–53. Springer, Cham, April 2017. ISBN 978-3-319-56880-5.

[PG11] U. Panniello and M. Gorgoglione. Context-aware recommender systems: A comparison of three approaches. In *Fifth International Workshop on New Challenges in Distributed Information Filtering and Retrieval*, volume 771, pp. 1–12. CEUR Workshop Proceedings, September 2011. ISSN 1613-0073.

[PG13] N. Polatidis and C. K. Georgiadis. Mobile recommender systems: An overview of technologies and challenges. In *Second International Conference on Informatics and Applications (ICIA)*, pp. 282–287. IEEE, September 2013. ISBN 978-1-4673-5256-7.

[PG17] N. Polatidis and C. K. Georgiadis. A dynamic multi-level collaborative filtering method for improved recommendations. *Computer Standards & Interfaces*, 51(Supplement C):14–21, 2017. ISSN 0920-5489.

[PGPM17] N. Polatidis, C. K. Georgiadis, E. Pimenidis, and H. Mouratidis. Privacy-preserving collaborative recommendations based on random perturbations. *Expert Systems with Applications*, 71:18–25, 2017. ISSN 0957-4174.

[PGPS15] N. Polatidis, C. K. Georgiadis, E. Pimenidis, and E. Stiakakis. A method for privacy-preserving context-aware mobile recommendations. In S. K.

Katsikas and A. B. Sideridis, editors, *Sixth International Conference on E-Democracy – Citizen Rights in the World of the New Computing Paradigms*, volume 570 of *Communications in Computer and Information Science (CCIS)*, pp. 62–74. Springer, Cham, December 2015. ISBN 978-3-319-27164-4.

[PGPS16] N. Polatidis, C. Georgiadis, E. Pimenidis, and E. Stiakakis. Privacy-preserving recommendations in context-aware mobile environments. *Information and Computer Security*, 25(1), 2016. ISSN 2056-4961.

[PHJ15] P. Pirasteh, D. Hwang, and J. J. Jung. Exploiting matrix factorization to asymmetric user similarities in recommendation systems. *Knowledge-Based Systems*, 83:51–57, 2015. ISSN 0950-7051.

[PHLG00] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory–and model–based approach. In *16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 473–480. Morgan Kaufmann Publishers Inc., San Francisco (USA), June 2000. ISBN 1-55860-709-9.

[PKCK12] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059–10072, 2012. ISSN 0957-4174.

[PKK06] S. Park, S. Kang, and Y.-K. Kim. A channel recommendation system in mobile environment. *IEEE Transactions on Consumer Electronics*, 52(1):33–39, February 2006. ISSN 0098-3063.

[PM06] G. Pitsilis and L. Marshall. A trust-enabled P2P recommender system. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 59–64. IEEE, June 2006. ISBN 0-7695-2623-3. ISSN 1524-4547.

[PM08] S. A. Petersen and J.-K. Markiewicz. PALLAS: Personalised language learning on mobile devices. In *Fifth IEEE International Conference on Wireless, Mobile, and Ubiquitous Technology in Education (WMUTE)*, pp. 52–59. IEEE, March 2008. ISBN 978-0-7695-3108-3.

[Pol17] N. Polatidis. *Recommendations in Mobile Commerce Environments: Supporting Quality and Privacy Requirements*. Phd thesis, University of Macedonia, School of Information Sciences, Department of Applied Informatics, Thessaloniki (Greece), July 2017.

[Por01] M. F. Porter. Snowball: A language for stemming algorithms, 2001. `http://snowball.tartarus.org/texts/introduction.html`, [Accessed on October 3, 2017].

[Pow11] D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, February 2011. ISSN 2229-3981.

[PPK05] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In P. Herrmann, V. Issarny, and S. Shiu, editors, *Third International Conference on Trust Management (iTrust)*, volume 3477 of *Lecture Notes in Computer Science (LNCS)*, pp. 224–239. Springer, Berlin, Heidelberg, May 2005. ISBN 978-3-540-32040-1.

[PRPT05] M. Papagelis, I. Rousidis, D. Plexousakis, and E. Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. In M.-S. Hacid, N. V. Murray, Z. W. Raś, and S. Tsumoto, editors, *15th International Symposium on Methodologies for Intelligent Systems (ISMIS)*, volume 3488 of *Lecture Notes in Computer Science (LNCS)*, pp. 553–561. Springer, Berlin, Heidelberg, May 2005. ISBN 978-3-540-31949-8.

[PTG08] 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, November 2008. ISSN 1041-4347.

[PTG+09] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- versus post-filtering approaches in context-aware recommender systems. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 265–268. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[PTG14a] U. Panniello, A. Tuzhilin, and M. Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1):35–65, February 2014. ISSN 1573-1391.

[PTG14b] U. Panniello, A. Tuzhilin, and M. Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity: Which contextual modeling, pre-filtering and post-filtering methods perform the best. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65, February 2014. ISSN 1573-1391.

[PYC06] H.-S. Park, J.-O. Yoo, and S.-B. Cho. A context-aware music recommendation system using fuzzy Bayesian networks with utility theory. In L. Wang, L. Jiao, G. Shi, X. Li, and J. Liu, editors, *Third International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 4223 of *Lecture Notes in Computer Science (LNCS)*, pp. 970–979. Springer, Berlin, Heidelberg, September 2006. ISBN 978-3-540-45917-0.

[PZC+14] T. Phan, J. Zhou, S. Chang, J. Hu, and J. Lee. Collaborative recommendation of photo-taking geolocations. In *Third ACM Multimedia Workshop on Geotagging and Its Applications in Multimedia (GeoMM)*, pp. 11–16. ACM, New York (USA), November 2014. ISBN 978-1-4503-3127-2.

[PZCG14] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, May 2014. ISSN 1553-877X.

[QLXL10] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, August 2010. ISSN 1573-7659.

[QZC+14] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, and L. Guo. Improving collaborative recommendation via location-based user-item subgroup. *Procedia Computer Science*, 29:400–409, 2014. ISSN 1877-0509.

[QZH+14] Z. Qiao, P. Zhang, J. He, Y. Cao, C. Zhou, et al. Combining geographical information of users and content of items for accurate rating prediction. In

23rd International Conference on World Wide Web Companion (WWW Companion), pp. 361–362. ACM, New York (USA), April 2014. ISBN 978-1-4503-2745-9.

[RA+94] R. Rakesh Agrawal, Srikant et al. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *20th International Conference on Very Large Data Bases (VLDB)*, volume 1215, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (USA), September 1994. ISBN 1-55860-153-8.

[Rab89] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 0018-9219.

[RC03] A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, 2003. ISSN 1617-4909.

[RCK+17] C. Ren, J. Chen, Y. Kuo, D. Wu, and M. Yang. Recommender system for mobile users. *Multimedia Tools and Applications*, pp. 1–21, 2017. ISSN 1573-7721.

[RDB+08] M. Roberts, N. Ducheneaut, B. Begole, K. Partridge, B. Price, et al. Scalable architecture for context-aware activity-detecting mobile recommendation systems. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6. IEEE, June 2008. ISBN 978-1-4244-2100-8.

[Ren10] S. Rendle. Factorization machines. In *10th International Conference on Data Mining (ICDM)*, pp. 995–1000. IEEE, December 2010. ISBN 978-0-7695-4256-0. ISSN 1550-4786.

[RG65] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, October 1965. ISSN 0001-0782.

[RGAQ17] A. Rasheed, S. Gillani, S. Ajmal, and A. Qayyum. Vehicular ad hoc network (VANET): A survey, challenges, and applications. In *Second International Workshop on Vehicular Ad-Hoc Networks for Smart Cities (IWVSC)*, volume 548 of *Advances in Intelligent Systems and Computing (AISC)*, pp. 39–51. Springer, Singapore, August 2017. ISBN 978-981-10-3503-6.

[RGFST11] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 635–644. ACM, New York (USA), July 2011. ISBN 978-1-4503-0757-4.

[RGGV14] X. Ramirez-Garcia and M. García-Valdez. Post-filtering for a restaurant context-aware recommender system. In O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk, editors, *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, volume 547 of *Studies in Computational Intelligence (SCI)*, pp. 695–707. Springer, Cham, 2014. ISBN 978-3-319-05169-7.

[RGJDSRDA09] J. A. Recio-Garcia, G. Jimenez-Diaz, A. A. Sanchez-Ruiz, and B. Diaz-Agudo. Personality aware recommendations to groups. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 325–328. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[RHS+13] T. Ruotsalo, K. Haav, A. Stoyanov, S. Roche, E. Fani, et al. SMARTMUSEUM: A mobile recommender system for the Web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 20:50–67, May 2013. ISSN 1570-8268.

[Ric10] F. Ricci. Mobile recommender systems. *Information Technology & Tourism*, 12(3):205–231, 2010. ISSN 1098-3058.

[Rip01] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *First International Conference on Peer-to-Peer Computing*, pp. 99–100. IEEE, August 2001. ISBN 0-7695-1503-7.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 175–186. ACM, New York (USA), October 1994. ISBN 0-89791-689-1.

[Ris01] I. Rish. An empirical study of the Naïve Bayes classifier. In *International Workshop on Empirical Methods in AI, in conjunction with the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 41–46. August 2001.

[RjHH09] S. Rho, B. jun Han, and E. Hwang. SVR-based music mood classification and context-based music recommendation. In *17th ACM International Conference on Multimedia (MM)*, pp. 713–716. ACM, New York (USA), October 2009. ISBN 978-1-60558-608-3.

[RK17] Y. S. Rawat and M. S. Kankanhalli. ClickSmart: A context-aware viewpoint recommendation system for mobile photography. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):149–158, January 2017. ISSN 1558-2205.

[RKGP17] R. Reddy, N. Keesara, V. Garg, and V. Pudi. Plug load identification using regression based nearest neighbor classifier. In *Eighth International Conference on Future Energy Systems (e-Energy)*, pp. 101–110. ACM, New York (USA), May 2017. ISBN 978-1-4503-5036-5.

[RKM+01] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, November 2001. ISSN 1089-7801.

[RM06] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer networks*, 50(17):3485–3521, 2006. ISSN 1389-1286.

[RMZ13] A. Rosi, M. Mamei, and F. Zambonelli. Integrating social sensors and pervasive services: Approaches and perspectives. *International Journal of Pervasive Computing and Communications*, pp. 294–310, 2013. ISSN 1742-7371.

[Rob06] C. M. Roberts. Radio frequency identification (RFID). *Computers & Security*, 25(1):18–26, February 2006. ISSN 0167-4048.

[Roh07] A. Rohatgi. WebPlotDigitizer, 2007. `http://arohatgi.info/WebPlotDigitizer/app`, [Accessed on October 3, 2017].

[RRSK11] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85820-3.

[RSHS17] X. Ren, M. Song, E. Haihong, and J. Song. Context-aware probabilistic matrix factorization modeling for Point-of-Interest recommendation. *Neurocomputing*, 241:38–55, 2017. ISSN 0925-2312.

[RV96] R. Real and J. M. Vargas. The probabilistic basis of Jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.

[RV97] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997. ISSN 0001-0782.

[RXM+13] I. Rubino, J. Xhembulla, A. Martina, A. Bottino, and G. Malnati. MusA: Using indoor positioning and navigation to enhance cultural experiences in a museum. *Sensors*, 13(12):17445–17471, 2013. ISSN 1424-8220.

[Sab16] A. Sabic. Proactive recommendation delivery. In *10th ACM Conference on Recommender Systems (RecSys)*, pp. 459–462. ACM, New York (USA), September 2016. ISBN 978-1-4503-4035-9.

[Sai16] T. L. Saito. SQLite JDBC Driver 3.14.2, 2016. `https://bitbucket.org/xerial/sqlite-jdbc`, [Accessed on October 3, 2017].

[SAQM17] N. Shah, S. A. Abid, D. Qian, and W. Mehmood. A survey of P2P content sharing in MANETs. *Computers & Electrical Engineering*, 57:55–68, January 2017. ISSN 0045-7906.

[Sat96] M. Satyanarayanan. Fundamental challenges in mobile computing. In *15th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 1–7. ACM, New York (USA), May 1996. ISBN 0-89791-800-2.

[Sat01] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001. ISSN 1070-9916.

[Sat10] M. Satyanarayanan. Mobile computing: The next decade. In *First ACM Workshop on Mobile Cloud Computing (MCS)*, pp. 5:1–5:6. ACM, New York (USA), June 2010. ISBN 978-1-4503-0155-8.

[SAW94] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 85–90. IEEE, December 1994.

[SB10] O. Santos and J. Boticario. Modeling recommendations for the educational domain. *Procedia Computer Science*, 1(2):2793–2800, September 2010. ISSN 1877-0509. Proceedings of the First Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL).

[SBCH12] N. S. Savage, M. Baranski, N. E. Chavez, and T. Höllerer. I'm Feeling LoCo: A location based context aware recommendation system. In G. Gartner and F. Ortag, editors, *Eighth International Symposium on Location-Based Services: Advances in Location-Based Services*, Lecture Notes in Geoinformation and Cartography (LNGC), pp. 37–54. Springer, Berlin, Heidelberg, November 2012. ISBN 978-3-642-24198-7.

[SBE+12] M. Sarwat, J. Bao, A. Eldawy, J. J. Levandoski, A. Magdy, et al. Sindbad: A location-based social networking system. In *ACM SIGMOD International Conference on Management of Data*, pp. 649–652. ACM, New York (USA), May 2012. ISBN 978-1-4503-1247-9.

[SBM12] S. Schelter, C. Boden, and V. Markl. Scalable similarity-based neighborhood methods with MapReduce. In *Sixth ACM Conference on Recommender Systems (RecSys)*, pp. 163–170. ACM, New York (USA), September 2012. ISBN 978-1-4503-1270-7.

[SBMD10] R. Schirru, S. Baumann, M. Memmel, and A. Dengel. Extraction of contextualized user interest profiles in social sharing platforms. *Journal of Universal Computer Science*, 16(16):2196–2213, August 2010.

[SC00] B. Smyth and P. Cotter. A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2–3):53–59, April 2000. ISSN 0950-7051.

[SES15] S. Saeedi and N. El-Sheimy. Activity recognition using fusion of low-cost sensors on a smartphone for mobile navigation application. *Micromachines*, 6(8):1100–1134, August 2015.

[SFHS07] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science (LNCS)*, chapter 9, pp. 291–324. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-72079-9.

[SFR06] K. Shyong, D. Frankowski, and J. Riedl. Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. In G. Müller, editor, *International Conference on Emerging Trends in Information and Communication Security (ETRICS)*, volume 3995 of *Lecture Notes in Computer Science (LNCS)*, pp. 14–29. Springer, Berlin, Heidelberg, June 2006. ISBN 978-3-540-34642-5.

[SG11] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 8, pp. 257–297. Springer, Boston, MA, 2011. ISBN 978-0-387-85820-3.

[SGM15] F. Shi, C. Ghedira, and J.-L. Marini. Context adaptation for smart recommender systems. *IT Professional*, 17(6):18–26, November 2015. ISSN 1520-9202.

[SGRB08] O. C. Santos, J. Granado, E. Raffenne, and J. G. Boticario. Offering recommendations in OpenACS/dotLRN. In *Seventh International Conference on Community based Environments*, pp. 37–46. 2008.

[SJM14] A. Sotsenko, M. Jansen, and M. Milrad. Using a rich context model for a news recommender system for mobile users. In *22nd Conference on User Modeling, Adaptation, and Personalization (UMAP)*, volume 1181, pp. 1–4. CEUR Workshop Proceedings, July 2014. ISSN 1613-0073.

[SK09] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4:2–4:2, January 2009. ISSN 1687-7470.

[SK11] N. Shabib and J. Krogstie. The use of data mining techniques in location-based recommender system. In *Seventh ACM International Conference on Web Intelligence, Mining and Semantics (WIMS)*, pp. 28:1–28:7. ACM, New York (USA), June 2011. ISBN 978-1-4503-0148-0.

[SK17] M. Sengaliappan and K. Kumaravel. Analysis study of wireless technology and its communication standards using IEEE 802.11. *Analysis*, 4(6):4061–4066, June 2017. ISSN 2350-0328.

[SKK04] I. Schwab, A. Kobsa, and I. Koychev. Learning user interests through positive examples using content analysis and collaborative filtering. *User Modeling and User-Adapted Interaction*, 14(5):469–475, 2004.

[SKKR99] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *First ACM Conference on Electronic Commerce (EC)*, pp. 158–167. ACM, New York (USA), November 1999. ISBN 1-58113-176-3.

[SKKR00] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Second ACM Conference on Electronic Commerce (EC)*, pp. 158–167. ACM, New York (USA), 2000. ISBN 1-58113-272-7.

[SKKR01] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web (WWW)*, pp. 285–295. ACM, New York (USA), May 2001. ISBN 1-58113-348-0.

[SKKR02] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental Singular Value Decomposition algorithms for highly scalable recommender systems. In *Second International Conference on Computer and Information Science (ICIS)*, pp. 27–28. Citeseer, August 2002.

[SKP+14] A. Smirnov, A. Kashevnik, A. Ponomarev, N. Shilov, and N. Teslya. Proactive recommendation system for m-tourism application. In B. Johansson, B. Andersson, and N. Holmberg, editors, *13th International Conference on Perspectives in Business Informatics Research (BIR)*, volume 194 of *Lecture Notes in Business Information Processing (LNBIP)*, pp. 113–127. Springer, Cham, September 2014. ISBN 978-3-319-11370-8.

[SKR01] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. In R. Kohavi and F. Provost, editors, *Applications of Data Mining to Electronic Commerce*, volume 5, pp. 115–153. Springer, Boston, MA, 2001. ISBN 978-1-4615-1627-9. A Special Issue of Data Mining and Knowledge Discovery.

[SKR05] B. M. Sarwar, J. A. Konstan, and J. T. Riedl. Distributed recommender systems for internet commerce. In M. Khosrow-Pour, editor, *Encyclopedia of Information Science and Technology*, chapter 159, pp. 907–911. Idea Group, first edition, 2005. ISBN 1-59140-553-X.

[SKS13] R. Sumbaly, J. Kreps, and S. Shah. The Big Data ecosystem at LinkedIn. In *ACM SIGMOD International Conference on Management of Data*, pp. 1125–1134. ACM, New York (USA), June 2013. ISBN 978-1-4503-2037-5.

[SKY11] M. Sirivianos, K. Kim, and X. Yang. Socialfilter: Introducing social trust to collaborative spam mitigation. In *2011 Proceedings IEEE INFOCOM*,

pp. 2300–2308. IEEE, April 2011. ISBN 978-1-4244-9921-2. ISSN 0743-166X.

[SL15] A. Sen and M. Larson. From sensors to songs: A learning-free novel music recommendation system using contextual sensor data. In *International Workshop on Location-Aware Recommendations (LocalRec) co-located, in conjunction with the Ninth ACM Conference on Recommender Systems (RecSys)*, volume 1405, pp. 40–43. CEUR Workshop Proceedings, September 2015. ISSN 1613-0073.

[SL17] B. Smith and G. Linden. Two decades of recommender systems at Amazon.com. *IEEE Internet Computing*, 21(3):12–18, May 2017. ISSN 1089-7801.

[SLEM14] M. Sarwat, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. LARS∗: An efficient and scalable location-aware recommender system. *IEEE Transactions on Knowledge and Data Engineering*, 26(6):1384–1399, 2014.

[SLG11] F. C. Seijo, J. M. F. Luna, and J. F. H. Guadix. *Recuperación de Información: un enfoque práctico y multidisciplinar*. Ra-Ma, 2011. ISBN 978-84-9964-112-6.

[SLH09] Y. Shi, M. Larson, and A. Hanjalic. Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering. In *Third ACM Conference on Recommender Systems (RecSys)*, pp. 125–132. ACM, New York (USA), October 2009. ISBN 978-1-60558-435-5.

[SLH10] Y. Shi, M. Larson, and A. Hanjalic. Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *International Workshop on Context-Aware Movie Recommendation (CAMRa)*, pp. 34–40. ACM, New York (USA), September 2010. ISBN 978-1-4503-0258-6.

[SLH13] Y. Shi, M. Larson, and A. Hanjalic. Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):16:1–16:19, February 2013. ISSN 2157-6904.

[SLH14] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3:1–3:45, May 2014. ISSN 0360-0300.

[SM86] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[SM95] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *SIGCHI International Conference on Human Factors in Computing Systems (CHI)*, pp. 210–217. ACM, New York (USA), May 1995. ISBN 0-201-84705-1.

[SMB07] J. J. Sandvig, B. Mobasher, and R. Burke. Robustness of collaborative recommendation based on association rule mining. In *ACM Conference on Recommender Systems (RecSys)*, pp. 105–112. ACM, New York, NY, USA, October 2007. ISBN 978-1-59593-730-8.

[SMP+11] G. Sielis, C. Mettouris, G. Papadopoulos, A. Tzanavari, R. Dols, et al. A context aware recommender system for creativity support tools. *Journal of Universal Computer Science*, 17(12):1743–1763, August 2011.

[SN99] I. Soboroff and C. Nicholas. Combining content and collaboration in text filtering. In *International Workshop on Machine Learning for Information Filtering, in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 99, pp. 86–91. August 1999.

[SNC13] A. Stanescu, S. Nagar, and D. Caragea. A hybrid recommender system: User profiling from keywords and ratings. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pp. 73–80. IEEE, Washington (USA), November 2013. ISBN 978-0-7695-5145-6.

[Son16] L. H. Son. Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 58:87–104, 2016. ISSN 0306-4379.

[SP14] A. Smirnov and A. Ponomarev. A hybrid peer-to-peer recommendation system architecture based on locality-sensitive hashing. In *15th Conference of Open Innovations Association FRUCT*, pp. 119–125. IEEE, April 2014. ISBN 978-5-8088-0890-4. ISSN 2305-7254.

[SPGR08] R. Schifanella, A. Panisson, C. Gena, and G. Ruffo. MobHinter: Epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *ACM International Conference on Recommender Systems (RecSys)*, pp. 27–34. ACM, New York (USA), October 2008. ISBN 978-1-60558-093-7.

[SPK04] M. V. Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application COMPASS. In P. M. E. D. Bra and W. Nejdl, editors, *Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, volume 3137 of *Lecture Notes in Computer Science (LNCS)*, pp. 235–244. Springer, Berlin, Heidelberg, August 2004. ISBN 978-3-540-27780-4.

[SPS13] N. Stojnic, L. Probst, and H. Schuldt. COMPASS-optimized routing for efficient data access in mobile chord-based P2P systems. In *14th International Conference on Mobile Data Management (MDM)*, volume 1, pp. 46–55. IEEE, June 2013. ISBN 978-0-7695-4973-6. ISSN 1551-6245.

[SPUP02] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 253–260. ACM, New York (USA), August 2002. ISBN 1-58113-561-0.

[SRF13] B. Shapira, L. Rokach, and S. Freilikhman. Facebook single and cross domain data for recommendation systems. *User Modeling and User-Adapted Interaction*, 23(2):211–247, April 2013. ISSN 1573-1391.

[SS14] J. Shah and L. Sahu. A survey of various hybrid based recommendation method. *International Journal of Advanced Research in Computer Science and Software Engineering*, 11(4):367–371, November 2014. ISSN 2277-128X.

[SS15] J. A. Singam and S. Srinivasan. Optimal keyword search for recommender system in Big Data application. *ARPN Journal of Engineering and Applied Sciences (ARPN-JEAS)*, 10(7):3243–3247, 2015. ISSN 1819-6608.

[SS17a] P. M. Sahu and D. M. Sable. Targeted advertising using location - based behavioral data & social data. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 5(VII):828–842, July 2017. ISSN 2321-9653.

[SS17b] P. M. Sahu and D. M. Sable. Targeted advertising using location by using FP Growth algorithm. *International Journal of Technical Research and Applications (IJTRA)*, 5(1):56–61, February 2017. ISSN 2320-8163.

[SSBRS16] O. C. Santos, M. Saneiro, J. G. Boticario, and M. Rodriguez-Sanchez. Toward interactive context-aware affective educational recommendations in computer assisted language learning. *New Review of Hypermedia and Multimedia*, 22(1-2):27–57, 2016.

[SSW04] D. O. Sullivan, B. Smyth, and D. Wilson. Preserving recommender accuracy and diversity in sparse datasets. *International Journal on Artificial Intelligence Tools*, 13(1):219–235, 2004.

[Sta08] Startup Spotify AB. Spotify, 2008. `https://www.spotify.com`, [Accessed on October 3, 2017].

[sta17a] statista. Number of apps available in leading app stores as of june 2016, 2017. `https://www.statista.com/statistics/276623`, [Accessed on October 3, 2017].

[sta17b] statista. Number of mobile phone users worldwide from 2013 to 2019, 2017. `https://www.statista.com/statistics/274774`, [Accessed on October 3, 2017].

[sta17c] statista. Number of smartphone users worldwide from 2014 to 2020, 2017. `https://www.statista.com/statistics/330695`, [Accessed on October 3, 2017].

[SU11] T. Sandholm and H. Ung. Real-time, location-aware collaborative filtering of web content. In *Workshop on Context-awareness in Retrieval and Recommendation (CaRR)*, pp. 14–18. ACM, New York (USA), February 2011. ISBN 978-1-4503-0625-6.

[SUPOK08] S. Sae-Ueng, S. Pinyapong, A. Ogino, and T. Kato. Personalized shopping assistance service at ubiquitous shop space. In *22nd International Conference on Advanced Information Networking and Applications-Workshops (AINAW)*, pp. 838–843. IEEE, March 2008.

[SYBA10] X. Shen, H. Yu, J. Buford, and M. Akon. *Handbook of Peer-to-Peer Networking*, volume 34. Springer, 2010. ISBN 978-0-387-09751-0.

[Sym16] P. Symeonidis. Matrix and tensor decomposition in recommender systems. In *10th ACM Conference on Recommender Systems (RecSys)*, pp. 429–430. ACM, New York (USA), September 2016. ISBN 978-1-4503-4035-9.

[SYPT10] J.-H. Su, H.-H. Yeh, S. Y. Philip, and V. S. Tseng. Music recommendation using content and context information mining. *IEEE Intelligent Systems*, 25(1):16–26, January 2010. ISSN 1541-1672.

[SZ11] B. Shapira and B. Zabar. Personalized search: Integrating collaboration and social networks. *Journal of the American Society for Information Science and Technology*, 62(1):146–160, 2011. ISSN 1532-2890.

[SZ17] P. Symeonidis and A. Zioupos. *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer, 2017. ISBN 978-3-319-41356-3.

[TBTL07] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 439–446. ACM, New York (USA), July 2007. ISBN 978-1-59593-597-7.

[TCL08] C.-Y. Tsai, S.-Y. Chou, and S.-W. Lin. Location-aware tour guide systems in museum. In R. Curran, S.-Y. Chou, and A. Trappey, editors, *15th ISPE International Conference on Concurrent Engineering (CE): Collaborative Product and Service Life Cycle Management for a Sustainable World*, Advanced Concurrent Engineering (ACENG), pp. 349–356. Springer, London, August 2008. ISBN 978-1-84800-972-1.

[TCL11] D. C.-E. Teng, N.-S. Chen, and C.-H. Lee. Enhancing english reading comprehension by integrating direct access to digital materials and scaffolded questionings in paper prints. In *11th IEEE International Conference on Advanced Learning Technologies (ICALT)*, pp. 244–248. IEEE, July 2011. ISBN 978-1-61284-209-7. ISSN 2161-3761.

[TGG+17] X. Teng, D. Guo, Y. Guo, X. Zhou, Z. Ding, et al. IONavi: An indoor-outdoor navigation service via mobile crowdsensing. *ACM Transactions on Sensor Networks (TOSN)*, 13(2):12:1–12:28, April 2017. ISSN 1550-4859.

[TGNO92] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. *SIGMOD Record*, 21(2):321–330, June 1992. ISSN 0163-5808.

[TH01] L. Terveen and W. Hill. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, 1(2001):487–509, 2001.

[THD04] D. A. Tran, K. A. Hua, and T. T. Do. A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133, January 2004. ISSN 0733-8716.

[The04] The Museum of Modern Art (MoMA). MoMA, 2004. `https://www.moma.org`, [Accessed on October 3, 2017].

[The16] The Museum of Modern Art (MoMA). MoMA Collection, 2016. `https://github.com/MuseumofModernArt/collection/tree/v1.12`, [Accessed on October 3, 2017].

[THK11] C.-C. Tuan, C.-F. Hung, and T.-C. Kuei. Location dependent collaborative filtering recommendation system. In *Third International Conference on Future Network Technologies (ICFN)*. 2011.

[TPNT09] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10(March):623–656, 2009.

[TR09] G. Tumas and F. Ricci. Personalized mobile city transport advisory system. In W. Höpken, U. Gretzel, and R. Law, editors, *International Conference on Information and Communication Technologies in Tourism*, pp. 173–183. Springer, Vienna (Austria), 2009. ISBN 978-3-211-93971-0.

[TRL+09] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, et al. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Seventh ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 85–98. ACM, New York (USA), November 2009. ISBN 978-1-60558-519-2.

[TS04] H.-W. Tung and V.-W. Soo. A personalized restaurant recommender agent for mobile e-service. In *2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE)*, pp. 259–262. March 2004. ISBN 0-7695-2073-1.

[TS05] Y. Takeuchi and M. Sugimoto. An outdoor recommendation system based on user location history. In H. Ko, A. Krüger, S.-G. Lee, and W. Woo, editors, *First International Workshop on Personalized Context Modeling and Management for UbiComp Applications (UbiPCMM)*, pp. 91–100. CEUR Workshop Proceedings, 2005. ISSN 1613-0073.

[TS06a] Y. Takeuchi and M. Sugimoto. CityVoyager: An outdoor recommendation system based on user location history. In J. Ma, H. Jin, L. T. Yang, and J. J. P. Tsai, editors, *Third International Conference on Ubiquitous Intelligence and Computing (UIC)*, volume 4159 of *Lecture Notes in Computer Science (LNCS)*, pp. 625–636. Springer, Berlin, Heidelberg, September 2006. ISBN 978-3-540-38092-4.

[TS06b] A. Turpin and F. Scholer. User performance versus precision measures for simple search tasks. In *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 11–18. ACM, New York (USA), 2006. ISBN 1-59593-369-7.

[TST06a] K. H. L. Tso and L. Schmidt-Thieme. Empirical analysis of attribute-aware recommendation algorithms with variable synthetic data. In A. F. Vladimir Batagelj, Hans-Hermann Bock and A. Žiberna, editors, *10th Jubilee Conference of the International Federation of Classification Societies: Data Science and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization (STUDIES CLASS), pp. 271–278. Springer, Berlin, Heidelberg, July 2006. ISBN 978-3-540-34416-2.

[TST06b] K. H. L. Tso and L. Schmidt-Thieme. Empirical analysis of attribute-aware recommender system algorithms using synthetic data. *Journal of Computers*, 1(4):18–29, 2006.

[TVC12] K.-N. Tran, D. Vatsalan, and P. Christen. ANU (Australian National University) online personal data generator and corruptor (GeCo), 2012. `http://dmm.anu.edu.au/geco`, [Accessed on October 3, 2017].

[TVC13] K.-N. Tran, D. Vatsalan, and P. Christen. GeCo: An online personal data generator and corruptor. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*, pp. 2473–2476. ACM, New York (USA), October 2013. ISBN 978-1-4503-2263-8.

[Tve01] A. Tveit. Peer-to-peer based recommendations for mobile commerce. In *First International Workshop on Mobile Commerce (WMC)*, pp. 26–29. ACM, New York (USA), 2001. ISBN 1-58113-376-6.

[TZAQL12] H. A. Tair, M. J. Zemerly, M. Al-Qutayri, and M. Leida. Architecture for context-aware pro-active recommender system. *International Journal Multimedia and Image Processing (IJMIP)*, 2(1/2):125–133, 2012.

[UBSR16] M. Unger, A. Bar, B. Shapira, and L. Rokach. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178, July 2016. ISSN 0950-7051.

[UF98] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *AAAI Workshop on Recommendation Systems, in conjunction with the 15th Conference on Artificial Intelligence*, volume 1, pp. 114–129. July 1998.

[Uni93] University of Waikato (New Zealand). Weka, 1993. `http://www.cs.waikato.ac.nz/ml/weka`, [Accessed on October 3, 2017].

[VC95] V. Vapnik and C. Cortes. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995. ISSN 1573-0565.

[VC11] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 109–116. ACM, New York (USA), October 2011. ISBN 978-1-4503-0683-6.

[VGGSPM11] B. Vargas-Govea, G. González-Serna, and R. Ponce-Medellın. Effects of relevant contextual features in the performance of a restaurant recommender system. In *Third Workshop on Context-Aware Recommender Systems (CARS), in conjunction with the Fifth ACM Conference on Recommender Systems (RecSys)*, volume 791. CEUR Workshop Proceedings, October 2011. ISSN 1613-0073.

[VHR12] D. G. Vico, G. Huecas, and J. S. Rodríguez. Generating context-aware recommendations using banking data in a mobile recommender system. In *Sixth International Conference on Digital Society (ICDS)*, pp. 73–78. IARIA, February 2012. ISBN 978-1-61208-176-2.

[VK16] R. Vignesh and D. Kavitha. A recommendation system with spam reduction based on clustering with on demand service. *International Research Journal of Engineering and Technology (IRJET)*, 3:959–966, November 2016. ISSN 2395-0056.

[VMO+12] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, et al. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4):318–335, October 2012. ISSN 1939-1382.

[Voo01] E. M. Voorhees. Evaluation by highly relevant documents. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 74–82. ACM, New York (USA), 2001. ISBN 1-58113-331-6.

[VPB+15] D. Véras, T. Prota, A. Bispo, R. Prudêncio, and C. Ferraz. A literature review of recommender systems in the television domain. *Expert Systems with Applications*, 42(22):9046–9076, December 2015. ISSN 0957-4174.

[VWB11] D. G. Vico, W. Woerndl, and R. Bader. A study on proactive delivery of restaurant recommendations for Android smartphones. In *International Workshop on Personalization in Mobile Applications (PeMA), in conjunction with the Fiftth ACM Conference on Recommender Systems (RecSys)*. October 2011.

[Wal96] C. Walck. Hand-book on statistical distributions for experimentalists. Technical report, University of Stockholm, 1996. Internal Report SUF–PFY/96–01.

[WBE09] W. Woerndl, M. Brocco, and R. Eigner. Context-aware recommender systems in mobile scenarios. *International Journal of Information Technology and Web Engineering (IJITWE)*, 4(1):67–85, 2009.

[WBS07] A. I. Wang, T. Bjornsgard, and K. Saxlund. Peer2Me–rapid application framework for mobile peer-to-peer applications. In *International Symposium on Collaborative Technologies and Systems (CTS)*, pp. 379–388. IEEE, May 2007. ISBN 978-0-9785699-1-4.

[WCH14] R. K. Wong, V. W. Chu, and T. Hao. Online role mining for context-aware mobile service recommendation. *Personal and Ubiquitous Computing*, 18(5):1029–1046, June 2014. ISSN 1617-4917.

[WCN12] Y. Wang, S. C.-F. Chan, and G. Ngai. Applicability of demographic recommender system to tourist attractions: A case study on TripAdvisor. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pp. 97–101. IEEE, Washington (USA), December 2012. ISBN 978-0-7695-4880-7.

[Wei13] G. Weiss. *Multiagent Systems*. MIT Press, Cambridge (USA), second edition, March 2013. ISBN 978-0-262-01889-0.

[WF15] F. Wortmann and K. Flüchter. Internet of Things. *Business & Information Systems Engineering*, 57(3):221–224, June 2015.

[WHBGV11] W. Woerndl, J. Huebner, R. Bader, and D. Gallego-Vico. A model for proactivity in mobile, context-aware recommender systems. In *Fifth ACM Conference on Recommender Systems (RecSys)*, pp. 273–276. ACM, New York, (USA), October 2011. ISBN 978-1-4503-0683-6.

[WHC+17] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69:29–39, March 2017. ISSN 0957-4174.

[WHF07] K. Wei, J. Huang, and S. Fu. A survey of e-commerce recommender systems. In *International Conference on Service Systems and Service Management*, pp. 1–5. IEEE, June 2007. ISBN 1-4244-0884-9. ISSN 2161-1890.

[Wil04] J. S. Wilson. *Sensor Technology Handbook*. Elsevier, 2004. ISBN 0-7506-7729-5.

[WL09] Y. Wang and F. Li. Vehicular ad hoc networks. In S. Misra, I. Woungang, and S. C. Misra, editors, *Guide to Wireless Ad Hoc Networks*, Computer Communications and Networks (CCN), chapter 20, pp. 503–525. Springer, London, 2009. ISBN 978-1-84800-328-6.

[WLX16] F. Wang, D. Li, and M. Xu. A location-aware TV show recommendation with localized sementaic analysis. *Multimedia Systems*, 22(4):535–542, July 2016. ISSN 1432-1882.

[WM05] C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, 30(1):79–82, December 2005.

[WMB07] C. A. Williams, B. Mobasher, and R. Burke. Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications*, 1(3):157–170, November 2007. ISSN 1863-2394.

[WPLR06] J. Wang, J. Pouwelse, R. L. Lagendijk, and M. J. T. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. In *ACM Symposium on Applied Computing (SAC)*, pp. 1026–1030. ACM, New York (USA), April 2006. ISBN 1-59593-108-2.

[WRW12] X. Wang, D. Rosenblum, and Y. Wang. Context-aware mobile music recommendation for daily activities. In *20th ACM International Conference on Multimedia (MM)*, pp. 99–108. ACM, New York (USA), 2012. ISBN 978-1-4503-1089-5.

[WS07] W. Woerndl and J. Schlichter. Introducing context into recommender systems. In *AAAI Workshop on Recommender Systems in E-commerce*, pp. 138–140. July 2007.

[WSS03] D. C. Wilson, B. Smyth, and D. O. Sullivan. Sparsity reduction in collaborative recommendation: A case-based approach. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(5):863–884, 2003.

[WSW07] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *IEEE 23rd International Conference on Data Engineering (ICDE)*, pp. 871–878. IEEE, April 2007. ISBN 978-1-4244-0832-0.

[WTH10] B. Wang, Z. Tao, and J. Hu. Improving the diversity of user-based top-N recommendation by cloud model. In *Fifth International Conference on Computer Science and Education (ICCSE)*, pp. 1323–1327. IEEE, August 2010. ISBN 978-1-4244-6005-2.

[WW11] S.-L. Wang and C.-Y. Wu. Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. *Expert Systems with applications*, 38(9):10831–10838, 2011.

[WYG+17] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, et al. Fog orchestration for Internet of Things services. *IEEE Internet Computing*, 21(2):16–24, March 2017. ISSN 1089-7801.

[WZH+12] J. Wang, C. Zeng, C. He, L. Hong, L. Zhou, et al. Context-aware role mining for mobile service recommendation. In *27th Annual ACM Symposium on Applied Computing (SAC)*, pp. 173–178. ACM, New York (USA), March 2012. ISBN 978-1-4503-0857-1.

[XA06] J. A. Xu and K. Araki. A SVM-based personal recommendation system for TV programs. In *12th International Multi-Media Modelling Conference*, pp. 401–404. IEEE, January 2006. ISBN 1-4244-0028-7. ISSN 1550-5502.

[XCH+10] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining*, pp. 211–222. SIAM, April 2010. ISBN 978-1-61197-280-1.

[XDL+16] Z. Xiang, S. Deng, S. Liu, B. Cao, and J. Yin. CAMER: A context-aware mobile service recommendation system. In *2016 IEEE International Conference on Web Services (ICWS)*, pp. 292–299. IEEE, June 2016. ISBN 978-1-5090-2675-3.

[XHHB02] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *22nd International Conference on Distributed Computing Systems*, pp. 363–371. IEEE, July 2002. ISBN 0-7695-1585-1. ISSN 1063-6927.

[XLY+05] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, et al. Scalable collaborative filtering using cluster-based smoothing. In *28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 114–121. ACM, New York (USA), August 2005. ISBN 1-59593-034-5.

[XYT+17] J. Xu, Y. Yao, H. Tong, X. Tao, and J. Lu. RaPare: A generic strategy for cold-start rating prediction problem. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1296–1309, June 2017. ISSN 1041-4347.

[XZYN16] J. Xiao, Z. Zhou, Y. Yi, and L. M. Ni. A survey on wireless indoor localization from the device perspective. *ACM Computing Surveys (CSUR)*, 49(2):1–31, June 2016. ISSN 0360-0300.

[XZZS17] J. Xu, Y. Zhong, W. Zhu, and F. Sun. Trust-based context-aware mobile social network service recommendation. *Wuhan University Journal of Natural Sciences*, 22(2):149–156, April 2017. ISSN 1993-4998.

[YAG17] A. M. Yagci, T. Aytekin, and F. S. Gurgen. Scalable and adaptive collaborative filtering by mining frequent item co-occurrences in a user feedback stream. *Engineering Applications of Artificial Intelligence*, 58:171–184, 2017. ISSN 0952-1976.

[Yah04] Yahoo! Research. Yahoo! Music Dataset, 2004. `http://webscope.sandbox.yahoo.com/catalog.php?datatype=rc`, [Accessed on October 3, 2017].

[YB07] L. Yujian and L. Bo. A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, June 2007. ISSN 0162-8828.

[YCC+15] H. Yin, B. Cui, L. Chen, Z. Hu, and C. Zhang. Modeling location-based user rating profiles for personalized recommendation. *ACM Transactions on Knowledge Discovery from Data*, 9(3):19:1–19:41, April 2015. ISSN 1556-4681.

[YCD08] W.-S. Yang, H.-C. Cheng, and J.-B. Dia. A location-aware recommender system for mobile shopping environments. *Expert Systems with Applications*, 34(1):437–445, January 2008. ISSN 0957-4174.

[YCS+14] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen. LCARS: A spatial item recommender system. *ACM Transactions on Information Systems*, 32(3):11:1–11:37, July 2014. ISSN 1046-8188.

[YD16] D. S. Yadav and P. K. Doke. Mobile cloud computing issues and solution framework. *International Research Journal of Engineering and Technology (IRJET)*, 03:1115–1118, November 2016. ISSN 2395-0056.

[Yel04] Yelp Fusion. Yelp, 2004. `http://www.yelp.com`, [Accessed on October 3, 2017].

[Yel14] Yelp. Yelp Dataset, 2014. `https://www.yelp.com/dataset_challenge`, [Accessed on October 3, 2017].

[YF94] J. Yang and D. Filo. Yahoo!, 1994. American Multinational Technology Company, `https://es.yahoo.com`, [Accessed on October 3, 2017].

[YH12] K. Yamamoto and T. Hayashi. Applying augmented reality to location-based restaurant recommendation for enhancing usability. In *13th Conference on Asia Pacific Industrial Engineering & Management Systems (APIEMS)*, pp. 126–129. December 2012.

[YH13] W.-S. Yang and S.-Y. Hwang. iTravel: A recommender system in mobile peer-to-peer environment. *Journal of Systems and Software*, 86(1):12–20, 2013. ISSN 0164-1212.

[YHL+15] L. Yang, F. Hao, S. Li, G. Min, H. Kim, et al. An efficient approach to generating location-sensitive recommendations in ad-hoc social network environments. *IEEE Transactions on Services Computing*, 8(3):520–533, May 2015. ISSN 1939-1374.

[YHSD14] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Parallel matrix factorization for recommender systems. *Knowledge and Information Systems*, 41(3):793–819, December 2014. ISSN 0219-3116.

[YJC07] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan. Challenges and approaches in large-scale P2P media streaming. *IEEE Multimedia*, 14(2):50–59, April 2007. ISSN 1070-986X.

[YK08] H. Yildirim and M. S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Second ACM Conference on Recommender Systems (RecSys)*, pp. 131–138. ACM, New York (USA), October 2008. ISBN 978-1-60558-093-7.

[YL98] J. Yen and R. Langari. *Fuzzy Logic: Intelligence, Control, and Information*. Prentice-Hall, Upper Saddle River (USA), first edition, 1998. ISBN 0-13-525817-0.

[YL10] Z. Yujie and W. Licai. Some challenges for context-aware recommender systems. In *5th International Conference on Computer Science and Education (ICCSE)*, pp. 362–365. IEEE, August 2010. ISBN 978-1-4244-6005-2.

[YLS+16] Ö. Yürür, C. H. Liu, Z. Sheng, V. C. M. Leung, W. Moreno, et al. Context-awareness for mobile sensing: A survey and future directions. *IEEE Communications Surveys & Tutorials*, 18(1):68–93, December 2016. ISSN 1553-877X.

[YLY+16] H. Yu, Y. Lian, S. Yang, L. Tian, and X. Zhao. Recommending features of mobile applications for developer. In J. Li, X. Li, S. Wang, J. Li, and Q. Z. Sheng, editors, *12th International Conference on Advanced Data Mining and Applications (ADMA)*, volume 10086 of *Lecture Notes in Computer*

Science (LNCS), pp. 361–373. Springer, Cham, December 2016. ISBN 978-3-319-49586-6.

[YLZC17] X.-P. Yang, H.-T. Lin, X.-G. Zhou, and B.-Y. Cao. Addition-min fuzzy relation inequalities with application in BitTorrent-like peer-to-peer file sharing system. *Fuzzy Sets and Systems*, 2017. ISSN 0165-0114. To appear.

[YMII14] R. Yus, E. Mena, S. Ilarri, and A. Illarramendi. SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Computing*, 15:87–99, December 2014. ISSN 1574-1192. Special Issue on Information Management in Mobile Applications.

[YNJ+07] Z. Yu, Y. Nakamura, S. Jang, S. Kajita, and K. Mase. Ontology-based semantic recommendation for context-aware e-learning. In J. Indulska, J. Ma, L. T. Yang, T. Ungerer, and J. Cao, editors, *Fourth International Conference on Ubiquitous Intelligence and Computing (UIC)*, volume 4611 of *Lecture Notes in Computer Science (LNCS)*, pp. 898–907. Springer, Berlin, Heidelberg, July 2007. ISBN 978-3-540-73549-6.

[YpC09] C.-C. Yu and H. ping Chang. Personalized location-based recommendation services for tour planning in mobile tourism applications. In T. D. Noia and F. Buccafurri, editors, *10th International Conference on E-Commerce and Web Technologies (EC-Web)*, volume 5692 of *Lecture Notes in Computer Science (LNCS)*, pp. 38–49. Springer, Berlin, Heidelberg, September 2009. ISBN 978-3-642-03964-5.

[YSZ17] A. Yadollahi, A. G. Shahraki, and O. R. Zaiane. Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys (CSUR)*, 50(2):25:1–25:33, May 2017. ISSN 0360-0300.

[YvLJ+17] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, et al. A new era for cities with fog computing. *IEEE Internet Computing*, 21(2):54–67, March 2017. ISSN 1089-7801.

[YWC15] Z. Yu, R. Wong, and C.-H. Chi. Efficient role mining for context-aware service recommendation using a high-performance cluster. *IEEE Transactions on Services Computing*, PP(99):1–14, October 2015. ISSN 1939-1374.

[YY10] K. F. Yeung and Y. Yang. A proactive personalized mobile news recommendation system. In *Developments in E-systems Engineering (DESE)*, pp. 207–212. IEEE, September 2010. ISBN 978-1-4244-8044-9.

[YYN10] K. F. Yeung, Y. Yang, and D. Ndzi. Context-aware news recommender in mobile hybrid P2P network. In *Second International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, pp. 54–59. IEEE, July 2010. ISBN 978-1-4244-7837-8.

[YYN12] K. F. Yeung, Y. Yang, and D. Ndzi. A proactive personalised mobile recommendation system using analytic hierarchy process and Bayesian network. *Journal of Internet Services and Applications*, 3(2):195–214, 2012. ISSN 1869-0238.

[YZYW13] D. Yang, D. Zhang, Z. Yu, and Z. Wang. A sentiment-enhanced personalized location recommendation system. In *24th ACM Conference on Hypertext and Social Media (HT)*, pp. 119–128. ACM, New York (USA), May 2013. ISBN 978-1-4503-1967-6.

[YZZ+06] Z. Yu, X. Zhou, D. Zhang, C.-Y. Chin, X. Wang, et al. Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5(3):68–75, 2006. ISSN 1536-1268.

[ZBM12] Y. Zheng, R. Burke, and B. Mobasher. Differential context relaxation for context-aware travel recommendation. In C. Huemer and P. Lops, editors, *13th International Conference on Electronic Commerce and Web Technologies (EC-Web)*, volume 123 of *Lecture Notes in Business Information Processing (LNBIP)*, pp. 88–99. Springer, Berlin, Heidelberg, September 2012. ISBN 978-3-642-32273-0.

[ZBM14] Y. Zheng, R. Burke, and B. Mobasher. Splitting approaches for context-aware recommendation: An empirical study. In *29th Annual ACM Symposium on Applied Computing (SAC)*, pp. 274–279. ACM, New York (USA), March 2014. ISBN 978-1-4503-2469-4.

[ZCZ+17] X. Zhou, L. Chen, Y. Zhang, D. Qin, L. Cao, et al. Enhancing online video recommendation using social user interactions. *The VLDB Journal*, 26(5):637–656, October 2017. ISSN 0949-877X.

[ZFY12] E. Zhong, W. Fan, and Q. Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SIAM International Conference on Data Mining*, pp. 744–755. SIAM, April 2012. ISBN 978-1-61197-282-5.

[ZGS16] X. Zou, M. Gonzales, and S. Saeedi. A context-aware recommendation system using smartphone sensors. In *Seventh Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1–6. October 2016. ISBN 978-1-5090-0996-1.

[ZGWW17] Y. Zhao, H. Gao, S. Wang, and F.-Y. Wang. A novel approach for traffic signal control: A recommendation perspective. *IEEE Intelligent Transportation Systems Magazine*, 9(3):127–135, 2017. ISSN 1939-1390.

[ZH08] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Second ACM Conference on Recommender Systems (RecSys)*, pp. 123–130. ACM, New York (USA), October 2008. ISBN 978-1-60558-093-7.

[Zha04] H. Zhang. The optimality of Naïve Bayes. In *17th Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pp. 562–567. AAAI Press, Menlo Park, California, May 2004. ISBN 1-57735-201-7.

[Zha05] H. Zhang. Exploring conditions for the optimality of Naïve Bayes. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(2):183–198, March 2005.

[Zhe13] Y. Zheng. The role of emotions in context-aware recommendation. In *Third Workshop on Human Decision Making in Recommender Systems (Decisions), in conjunction with the Seventh ACM Conference on Recommender Systems (RecSys)*, volume 1050. CEUR Workshop Proceedings, October 2013. ISSN 1613-0073.

[Zhe15a] Y. Zheng. CARSKit, 2015. `https://github.com/irecsys/CARSKit`, [Accessed on October 3, 2017].

[Zhe15b] Y. Zheng. Context suggestion: Solutions and challenges. In *IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 1602–1603. IEEE, November 2015. ISBN 978-1-4673-8493-3. ISSN 2375-9259.

[ZHW+10] J. Zhan, C.-L. Hsieh, I.-C. Wang, T.-S. Hsu, C.-J. Liau, et al. Privacy-preserving collaborative recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(4):472–476, July 2010. ISSN 1094-6977.

[ZI02] T. Zhang and V. S. Iyengar. Recommender systems using linear classifiers. *Journal of Machine Learning Research*, 2(February):313–334, 2002.

[ZI16] X. Zuo and A. Iamnitchi. A survey of socially aware peer-to-peer systems. *ACM Computing Surveys (CSUR)*, 49(1):1–28, May 2016. ISSN 0360-0300.

[Zie04] C.-N. Ziegler. Book-Crossing Dataset, 2004. `http://www2.informatik.uni-freiburg.de/~cziegler/BX`, [Accessed on October 3, 2017].

[ZJL+16] H. Zou, H. Jiang, Y. Luo, J. Zhu, X. Lu, et al. BlueDetect: An iBeacon-Enabled scheme for accurate and energy-efficient indoor-outdoor detection and seamless location-based service. *Sensors*, 16(2):268, February 2016.

[ZKL+10] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, et al. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

[ZL09] R. Zafarani and H. Liu. Flixster Dataset, 2009. `http://socialcomputing.asu.edu/datasets/Flixster`, [Accessed on October 3, 2017].

[ZLL11] Y. Zhang, H. Liu, and S. Li. A distributed collaborative filtering recommendation mechanism for mobile commerce based on cloud computing. *Journal of Information & Computational Science*, 8(16):3883–3891, 2011.

[ZLWL09] Q. Zhang, Y. Luo, C. Weng, and M. Li. A trust-based detecting mechanism against profile injection attacks in recommender systems. In *Third IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI)*, pp. 59–64. IEEE, July 2009. ISBN 978-0-7695-3758-0.

[ZM06] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys (CSUR)*, 38(2):1–6, July 2006. ISSN 0360-0300.

[ZMB14] Y. Zheng, B. Mobasher, and R. Burke. Context recommendation using multi-label classification. In *International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pp. 288–295. IEEE, August 2014. ISBN 978-1-4799-4143-8/14.

[ZMB15] Y. Zheng, B. Mobasher, and R. Burke. CARSKit: A Java-based context-aware recommendation engine. In *IEEE International Conference on Data Mining Workshop (ICDMW)*, pp. 1668–1671. IEEE, November 2015. ISBN 978-1-4673-8493-3. ISSN 2375-9259.

[ZMKL05] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *14th International Conference on World Wide Web (WWW)*, pp. 22–32. ACM, New York (USA), May 2005. ISBN 1-59593-046-9.

[ZOON12] T. Zuva, O. O. Olugbara, S. O. Ojo, and S. M. Ngwira. Image content in location-based shopping recommender systems for mobile users. *Advanced Computing: An International Journal (ACIJ)*, 3(4):1–8, July 2012.

[ZR08] R. Zhou and K. Rechert. Personalization for location-based e-learning. In *Second International Conference on Next Generation Mobile Applications, Services, and Technologies (NGMAST)*, pp. 247–253. IEEE, September 2008. ISBN 978-0-7695-3333-9. ISSN 2161-2889.

[ZS10] Z.-D. Zhao and M.-S. Shang. User-based collaborative-filtering recommendation algorithms on Hadoop. In *Third International Conference on Knowledge Discovery and Data Mining (WKDD)*, pp. 478–481. IEEE, January 2010. ISBN 978-1-4244-5398-6.

[ZSHM07] M. Zuckerberg, E. Saverin, C. Hughes, and D. Moskovitz. Facebook, 2007. `https://www.facebook.com`, [Accessed on October 3, 2017].

[ZSHM10] M. Zuckerberg, E. Saverin, C. Hughes, and D. Moskovitz. Facebook Places, 2010. `https://www.facebook.com/places`, [Accessed on October 3, 2017].

[ZSQJ12] Y. C. Zhang, D. Séaghdha, D. Quercia, and T. Jambor. Auralist: Introducing serendipity into music recommendation. In *Fifth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 13–22. ACM, New York (USA), February 2012. ISBN 978-1-4503-0747-5.

[ZWF13] W. Zhang, J. Wang, and W. Feng. Combining latent factor model with location features for event-based group recommendation. In *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 910–918. ACM, New York (USA), August 2013. ISBN 978-1-4503-2174-7.

[ZWSP08] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the Netflix Prize. In R. Fleischer and J. Xu, editors, *Fourth International Conference on Algorithmic Applications in Management (AAIM)*, volume 5034 of *Lecture Notes in Computer Science (LNCS)*, pp. 337–348. Springer, Berlin, Heidelberg, June 2008. ISBN 978-3-540-68880-8.

[ZWW+11] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, et al. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, December 2011. ISSN 1524-9050.

[ZZX13] D. Zhang, Q. Zou, and H. Xiong. CRUC: Cold-start rrecommendations using collaborative filtering in Internet of Things. *CoRR*, abs/1306.0165:1–8, June 2013.