Francesco Ricci · Lior Rokach
Bracha Shapira *Editors*

Recommender Systems Handbook

Second Edition

Springer

Editors
Francesco Ricci
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy

Lior Rokach
Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, Israel

Bracha Shapira
Ben-Gurion University of the Negev
Beer-Sheva, Israel

ISBN 978-1-4899-7636-9 ISBN 978-1-4899-7637-6 (eBook)
DOI 10.1007/978-1-4899-7637-6

Library of Congress Control Number: 2015953226

Springer New York Heidelberg Dordrecht London

Printed on acid-free paper

Springer Science+Business Media LLC New York is part of Springer Science+Business Media (www.springer.com)

Contents

Contributors

Gediminas Adomavicius
Department of Information and Decision Sciences, University of Minnesota, Minneapolis, MN, USA

Xavier Amatriain
Netflix, Los Gatos, CA, USA

Quora, Mountain View, USA

Justin Basilico
Netflix, Los Gatos, CA, USA

Gleb Beliakov
School of Information Technology, Deakin University, Burwood, VIC, Australia

Robert Bell
AT&T Labs – Research, Middletown, NJ, USA

Shlomo Berkovsky
CSIRO, Sydney, NSW, Australia

Dmitry Bogdanov
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

Peter Briggs
HeyStaks Technologies Ltd., NovaUCD, University College Dublin, Dublin, Ireland

Robin Burke
School of Computer Science, Telecommunication and Information Systems, DePaul University, Chicago, IL, USA

Tomasa Calvo
Departamento de Ciencias de la Computación, Universidad de Alcalá, Madrid, Spain

Iván Cantador
Universidad Autónoma de Madrid, Madrid, Spain

Pablo Castells
Universidad Autonoma de Madrid, Madrid, Spain

Li Chen
Hong Kong Baptist University, Hong Kong, China

Maurice Coyle
HeyStaks Technologies Ltd., NovaUCD, University College Dublin, Dublin, Ireland

Paolo Cremonesi
Politecnico di Milano, Milan, Italy

Marco de Gemmis
Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Christian Desrosiers
Software Engineering and IT Department, École de Technologie Supérieure, Montreal, QC, Canada

Hendrik Drachsler
Welten Institute Research Centre for Learning, Teaching and Technology, Open University of the Netherlands, Heerlen, The Netherlands

Mehdi Elahi
Free University of Bozen-Bolzano, Bolzano, Italy

Alexander Felfernig
University of Graz, Graz, Austria

Ignacio Fernández-Tobías
Universidad Autónoma de Madrid, Madrid, Spain

Arik Friedman
NICTA, Sydney, NSW, Australia

Gerhard Friedrich
Alpen-Adria-Universitaet Klagenfurt, Klagenfurt, Austria

Ulrike Gretzel
University of Queensland, Brisbane, QLD, Australia

Asela Gunawardana
Microsoft Research, Redmond, WA, USA

Ido Guy
Yahoo Labs, Haifa, Israel

Neil J. Hurley
Insight Centre for Data Analytics, School of Computer Science and Informatics, University College Dublin, Dublin, Ireland

Simon James
School of Information Technology, Deakin University, Burwood, VIC, Australia

Anthony Jameson
DFKI, German Research Center for Artificial Intelligence, Saarbrücken, Germany

Dietmar Jannach
TU Dortmund, Dortmund, Germany

Marius Kaminskas
Insight Centre for Data Analytics, University College Cork, Cork, Ireland

Dain Kaplan
Tokyo Institute of Technology, Tokyo, Japan

George Karypis
Computer Science & Engineering Department, University of Minnesota, Minneapolis, MN, USA

Peter Knees
Department of Computational Perception, Johannes Kepler University Linz, Linz, Austria

Bart P. Knijnenburg
Clemson University, Clemson, SC, USA

Irena Koprinska
School of Information Technologies, University of Sydney, Sydney, NSW, Australia

Yehuda Koren
Google Research, Mountain View, CA, USA

YoungOk Kwon
Sookmyung Women's University, Yongsan-gu, Seoul, Korea

Neal Lathia
Computer Laboratory, University of Cambridge, Cambridge, UK

Pasquale Lops
Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Nikos Manouselis
Agro-Know, Vrilissia, Greece

Luc Martens
iMinds - Ghent University, Ghent, Belgium

Judith Masthoff
University of Aberdeen, Aberdeen, UK

Brian McFee
Center for Data Science, New York University, New York, NY, USA

Kevin McNally
Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland

Cataldo Musto
Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Fedelucio Narducci
Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Xia Ning
Computer Science Department, Purdue University, West Lafayette, IN, USA

Michael P. O'Mahony
Insight Centre for Data Analytics, School of Computer Science and Informatics, University College Dublin, Dublin, Ireland

Josep M. Pujol
Cliqz, Munich, Germany

Francesco Ricci
Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy

Lior Rokach
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Neil Rubens
University of Electro-Communications, Tokyo, Japan

Olga C. Santos
aDeNu Research Group, UNED, Madrid, Spain

Markus Schedl
Department of Computational Perception, Johannes Kepler University Linz, Linz, Austria

Giovanni Semeraro
Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

Guy Shani
Information Systems Engineering, Ben Gurion University, Beer Sheva, Israel

Bracha Shapira
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Barry Smyth
Insight Centre for Data Analytics, University College Dublin, Dublin, Ireland

Masashi Sugiyama
Tokyo Institute of Technology, Tokyo, Japan

Nava Tintarev
University of Aberdeen, Aberdeen, UK

Marko Tkalcic
Johannes Kepler University, Linz, Austria

Alexander Tuzhilin
Department of Information, Operations and Management Sciences, Stern School of Business, New York University, New York, NY, USA

Kris Vanhecke
iMinds - Ghent University, Ghent, Belgium

Saúl Vargas
Universidad Autonoma de Madrid, Madrid, Spain

Katrien Verbert
Department of Computer Science, KU Leuven, Leuven, Belgium

Department of Computer Science, Vrije Universiteit Brussel, Brussel, Belgium

Martijn C. Willemsen
Eindhoven University of Technology, Eindhoven, The Netherlands

Kalina Yacef
School of Information Technologies, University of Sydney, Sydney, NSW, Australia

Kyung-Hyan Yoo
William Paterson University, Wayne, NJ, USA

Markus Zanker
Alpen-Adria-Universitaet Klagenfurt, Klagenfurt, Austria

Chapter 1
Recommender Systems: Introduction and Challenges

Francesco Ricci, Lior Rokach, and Bracha Shapira

1.1 Introduction

Recommender Systems (RSs) are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user [17, 41, 42]. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

"Item" is the general term used to denote what the system recommends to users. An RS normally focuses on a specific type of item (e.g., CDs or news) and, accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

RSs are primarily directed toward individuals who lack the sufficient personal experience or competence in order to evaluate the potentially overwhelming number of alternative items that a website, for example, may offer [42]. A prime example is a book recommender system that assists users in selecting a book to read. On the popular website, Amazon.com, the site employs an RS to personalize the online store for each customer [32]. Since recommendations are usually personalized, different users or user groups benefit from diverse, tailored suggestions. In addition, there are also non-personalized recommendations. These are much simpler to generate and are normally featured in magazines or newspapers. Typical examples

F. Ricci (✉)
Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
e-mail: fricci@unibz.it

L. Rokach • B. Shapira
Department of Information Systems Engineering,
Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: liorrk@bgu.ac.il; bshapira@bgu.ac.il

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
DOI 10.1007/978-1-4899-7637-6_1

include the top ten selections of books, CDs etc. While they may be useful and effective in certain situations, these types of non-personalized recommendations are not typically addressed by RS research.

In their simplest form, personalized recommendations are offered as ranked lists of items. In performing this ranking, RSs try to predict what the most suitable products or services are, based on the user's preferences and constraints. In order to complete such a computational task, RSs collect information from users regarding their preferences which are either explicitly expressed, e.g., as ratings for products or are inferred by interpreting the actions of the user. For instance, an RS may consider the navigation to a particular product page as an implicit sign of preference for the items shown on that page.

The development of RSs initiated from a rather simple observation: individuals often rely on recommendations provided by others in making routine, daily decisions [41, 51]. For example, it is common to rely on what one's peers recommend when selecting a book to read; employers count on recommendation letters in their recruiting decisions; and when selecting a movie to watch, individuals tend to read and rely on the movie reviews that a film critic has written, which appear in the newspaper they read.

In seeking to mimic this behavior, the first RSs applied algorithms in order to leverage recommendations produced by a community of users and deliver these recommendations to an "active" user, or a user looking for suggestions. The recommendations were for items that similar users, or those with similar tastes, had liked. This approach is termed collaborative-filtering and its rationale follows that if the active user agreed in the past with certain users, then the other recommendations coming from these similar users should be relevant as well as of interest to the active user.

As e-commerce websites began to develop, a pressing need emerged for providing recommendations derived from filtering the whole range of available alternatives. Users found it difficult to arrive at the most appropriate choices from the immense variety of items (products and services) that these websites offered.

The explosive growth and variety of information available on the Web and the rapid introduction of new e-business services (selling products, product comparison, auctions, etc.) frequently overwhelmed users, leading them to make poor decisions. The availability of choices, instead of producing a benefit, started to decrease users' well-being. It was understood that while choice is good, more choice is not always better. Indeed, choice, with its implications of freedom, autonomy, and self-determination can become excessive, and ultimately create a sense that freedom may come to be regarded as a kind of misery-inducing tyranny [49].

In recent years, RSs have proven to be a valuable means of coping with the information overload problem. Ultimately an RS addresses this phenomenon by pointing a user toward new, not-yet-experienced items that may be relevant to the user's current task. Upon a user's request, which can be articulated depending on the recommendation approach by the user's context and need, RSs generate recommendations using various types of knowledge and data about users, the available items, and previous transactions stored in customized databases. The user

can then browse the recommendations. One may accept them or not and may provide, immediately or at a later stage, implicit or explicit feedback. This user action and feedback can be stored in the recommender database and may be used for generating new recommendations in the coming user-system interactions.

As previously noted, the study of recommender systems is relatively new compared to research in other classical information system tools and techniques (e.g., databases or search engines). Recommender systems emerged as an independent research area in the mid-1990s [7, 24, 41, 51]. In recent years, the interest in recommender systems has dramatically increased, as the following facts indicate:

1. Recommender systems play an important role in highly-rated Internet sites such as Amazon.com, YouTube, Netflix, Spotify, LinkedIn, Facebook, Tripadvisor, Last.fm, and IMDb. Moreover many media companies are now developing and deploying RSs as part of the services they provide to their subscribers. For example, Netflix, the online provider of on-demand streaming media, awarded a million dollar prize to the team that first succeeded in substantially improving the performance of its recommender system [31].
2. There are conferences and workshops dedicated specifically to the field, namely the Association of Computing Machinery's (ACM) Conference Series on Recommender Systems (RecSys), established in 2007. This conference stands as the premier annual event in recommender technology research and applications. In addition, sessions dedicated to RSs are frequently included in more traditional conferences in the area of databases, information systems and adaptive systems. Additional noteworthy conferences within this scope include: ACM's Special Interest Group on Information Retrieval (SIGIR); User Modeling, Adaptation and Personalization (UMAP); Intelligent User Interfaces (IUI); World Wide Web (WWW); and ACM's Special Interest Group on Management Of Data (SIGMOD).
3. At institutions of higher education around the world, undergraduate and graduate courses are now dedicated entirely to RSs, tutorials on RSs are very popular at computer science conferences, and a book introducing RSs techniques has been published as well [27]. Springer is publishing several books on specific topics in recommender systems in its series: Springer Briefs in Electrical and Computer Engineering. A large, new collection of articles dedicated to recommender systems applications to software engineering has also recently been published [46].
4. There have been several special issues in academic journals which cover research and developments in the RSs field. Among the journals that have dedicated issues to RSs are: AI Communications (2008); IEEE Intelligent Systems (2007); International Journal of Electronic Commerce (2006); International Journal of Computer Science and Applications (2006); ACM Transactions on Computer Human Interaction (2005); ACM Transactions on Information Systems (2004); User Modeling and User-Adapted Interaction (2014, 2012); ACM Transactions on Interactive Intelligent Systems (2013); and ACM Transactions on Intelligent Systems and Technology (2015).

In this introductory chapter, we briefly discuss basic RS ideas and concepts. Our main goal is not to present a self-contained comprehensive survey on RSs but rather to delineate, in a coherent and structured way, the chapters included in this handbook and to help the reader navigate the rich and detailed content that the handbook offers. The reader can also consult these recent introductions or surveys on recommender systems [13, 30, 34, 40, 44]. At the end of this chapter, we have identified some research challenges that we believe are particularly important for the future of the area.

The handbook is divided into five sections: recommendation techniques; recommender systems evaluation; recommender systems applications; recommender systems and human computer interaction; and advanced algorithms.

The first section presents the techniques most popularly used today for building RSs, such as collaborative filtering; content-based, data mining methods; and context-aware methods.

The second section surveys techniques and approaches that have been utilized to evaluate the quality of the recommendations. The section also considers aspects that may affect RS design (domain, device, interfaces, users, etc.). Finally, it discusses methods, challenges and measures to be applied in evaluating the developed systems with user experiments.

The third section includes papers dealing with a number of issues related to how recommendations are presented, browsed, explained and visualized. Among them, this section focuses on user's privacy and the decision making process supported by a recommender system.

The fourth section is fully dedicated to applications of recommender systems. We offer here a broad spectrum of the usage of these techniques in music, mobile computing, dating, social networks, education, and movies.

The last section presents papers on various advanced topics, such as: the exploitation of active learning principles to guide the acquisition of new knowledge; novelty and diversity in the recommendations; suitable techniques for protecting a recommender system against attacks of malicious users; and RSs that aggregate multiple types of user feedback and preferences to build more reliable recommendations.

1.2 Recommender Systems' Function

In the previous section, we defined RSs as software tools and techniques that provide users with suggestions for items that a user may wish to utilize. Now we wish to refine this definition to illustrate a range of possible roles that an RS can play. Firstly, we must distinguish between the role played by the RS on behalf of the service provider, from that of the user of the RS. For instance, a travel recommender system is typically introduced by a travel intermediary such as Expedia.com, or a destination management organization, such as Visitfinland.com, in order to increase its turnover or sell more hotel rooms in the case of Expedia, and increase the number

of tourists to the destination in the case of the destination management organization [14, 43]. The user's primary motivations for accessing the two systems would be to find a suitable hotel and interesting events or attractions when visiting a destination.

In fact, there are various reasons as to why service providers may want to exploit this technology:

- *Increase the number of items sold.* This is probably the most important function for a commercial RS, i.e., to be able to sell an additional set of items compared to those usually sold without any kind of recommendation. This goal is achieved because the recommended items are likely to suit the user's needs and wants. Presumably the user will recognize this after having tried several recommendations.[1] Non-commercial applications have similar goals, even if there is no cost for the user that is associated with selecting an item. For instance, a content network aims at increasing the number of news items read on its site. In general, we can say that from the service provider's point of view, the primary goal for introducing an RS is to increase the conversion rate, i.e., the number of users that accept the recommendation and consume an item, compared to the number of simple visitors that just browse through the information.
- *Sell more diverse items.* Another major function of an RS is to enable the user to select items that might be hard to find without a precise recommendation. For instance, in a tourist RS the service provider is interested in promoting all the places of interest in a tourist area, not just the most popular ones. This could be difficult without an RS since the service provider cannot afford the risk of advertising places that are not likely to suit a particular user's taste. Therefore, an RS suggests or advertises unpopular places to the right users.
- *Increase the user satisfaction.* A well designed RS can also improve the experience of the user with the site or the application. The user will find the recommendations interesting, relevant and, with a properly designed human-computer interaction, he or she will also enjoy using the system. The combination of effective, accurate recommendations and a usable interface will increase the user's subjective evaluation of the system. This, in turn, will increase system usage and the likelihood that the recommendations will be accepted.
- *Increase user fidelity.* A user should be loyal to a website which, when visited, recognizes the old customer and treats him as a valued visitor. This is a standard feature of an RS since many RSs compute recommendations, thus leveraging the information acquired from the user during previous interactions such as the user's ratings of items. Consequently, the longer the user interacts with the site, the more refined the user's model becomes: the system's representation of the user's preferences develops and the effectiveness of the recommender output to customize and match to the user's preferences is increased.

[1] This issue, convincing the user to accept a recommendation, is discussed again when we explain the difference between predicting the user interest in an item and the likelihood that the user will select the recommended item.

- *Better understanding of what the user wants.* Another important function of an RS which can be leveraged to many other applications is the description of the user's preferences, which are collected either explicitly or predicted by the system. The service provider may then decide to reuse this knowledge for a number of other goals, such as improving the management of the item's stock or production. For instance, in the travel domain, destination management organizations can decide to advertise a specific region to new customer sectors or advertise a particular type of promotional message derived by analyzing the data collected by the RS (transactions of the users).

We mentioned above some important motivations as to why e-service providers introduce RSs. But users also may want an RS if it will effectively support their tasks or goals. Consequently an RS must balance the needs of these two players and offer a service that is valuable to both.

Herlocker et al. [26], in a paper that has become a classical reference in this field, define eleven popular tasks that an RS can assist in implementing. Some may be considered as the main or core tasks that are normally associated with an RS, such as offering suggestions for items that may be useful to a user. Others might be considered as more "opportunistic" ways to exploit an RS. As a matter of fact, this task differentiation is very similar to what happens with a search engine. Its primary function is to locate documents that are relevant to the user's information need, but it can also be used to check the importance of a webpage (looking at the position of the page in the result list of a query) or to discover the various usages of a word in a collection of documents.

- *Find Some Good Items:* Recommend to a user some items as a ranked list along with predictions of how much the user would like them (e.g., on a scale of one-to-five stars). This is the main recommendation task that many commercial systems address (see, for instance, Chap. 11). Some systems do not show the predicted rating.
- *Find all good items:* Recommend all the items that can satisfy some user needs. In such cases it is insufficient to just find some good items. This is especially true when the number of items is relatively small or when the RS is mission-critical, such as in medical or financial applications. In these situations, in addition to the benefit derived from carefully examining all the possibilities, the user may also benefit from the RS ranking of these items or from additional explanations that the RS generates.
- *Annotation in context:* Given an existing context, e.g., a list of items, emphasize some of them depending on the user's long-term preferences. For example, a TV recommender system might annotate which TV shows displayed in the electronic program guide (EPG) are worth watching (Chap. 15 provides interesting examples of this task).
- *Recommend a sequence:* Instead of focusing on the generation of a single recommendation, the idea is to recommend a sequence of items that is pleasing as a whole. Typical examples include recommending a TV series, a book on RSs after having recommended a book on data mining, or a compilation of musical tracks [28].

- *Recommend a bundle:* Suggest a group of items that fits well together. For instance, a travel plan may be composed of various attractions, destinations, and accommodation services that are located in a delimited area. From the point of view of the user, these various alternatives can be considered and selected as a single travel destination [45].
- *Just browsing:* In this task, the user browses the catalog without any imminent intention of purchasing an item. The task of the recommender is to help the user to browse the items that are more likely to fall within the scope of the user's interests for that specific browsing session. This is a task that has also been supported by adaptive hypermedia techniques [16].
- *Find credible recommender:* Some users do not trust recommender systems, thus they play with them to see how good they are at making recommendations. Hence, a certain system may also offer specific functions to let the users test its behavior in addition to those just required for obtaining recommendations.
- *Improve the profile:* This relates to the capability of the user to provide (input) information to the recommender system about what he or she likes and dislikes. This is a fundamental task that is strictly necessary to provide personalized recommendations. If the system has no specific knowledge about the active user, then it can only provide the same recommendations that would be delivered to an "average" user.
- *Express self:* Some users may not care about the recommendations at all. Rather, what is important to them is that they be allowed to contribute with their ratings and express their opinions and beliefs. The user satisfaction for that activity can still act as leverage, resulting in the user's continued loyalty to the application (as we mentioned prior, in discussing the service provider's motivations).
- *Help others:* Some users are happy to contribute with information, e.g., their evaluation of items (ratings), because they believe that the community benefits from their contribution. This could be a major motivation for entering information into a recommender system that is not used routinely. For instance, with an automobile RS, a user who has already purchased a new car is aware that the rating entered in the system is more likely to be useful to other users rather than to oneself, the next time a new-car-purchase is contemplated.
- *Influence others:* In Web-based RSs, there are users whose main goal is to explicitly influence other users into purchasing particular products. As a matter of fact, there are also malicious users that may use the system simply to promote or penalize certain items (see Chap. 28).

As these various points indicate, the role of an RS within an information system can be quite diverse. This diversity calls for the exploitation of a range of different knowledge sources and techniques. In the next two sections, we discuss the data that an RS manages and the core technique used to identify the right recommendations.

1.3 Data and Knowledge Sources

RSs are information processing systems that actively gather various kinds of data in order to build their recommendations. Data is primarily about the items to suggest and the users who will receive these recommendations. But, since the data and knowledge sources available for recommender systems can be very diverse, ultimately, whether it can be exploited or not depends on the recommendation technique (see also Sect. 1.4). This will become clearer in the various chapters included in this handbook.

In general, there are recommendation techniques that are knowledge-poor, namely, that use very simple and basic data, such as user ratings or evaluations for items (Chaps. 2 and 3). Other techniques are much more knowledge-dependent, in that they use ontological descriptions of the users or the items (Chap. 4), constraints (Chap. 5), or social relations and activities of the users (Chaps. 15 and 17). In any case, as a general classification, data used by RSs refers to three kinds of objects: items, users, and transactions, that is, relations between the users and the items.

Items Items are the objects that are recommended. Items may be characterized by their complexity and their value or utility. The value of an item may be positive if the item is useful to the user, or negative if the item is not appropriate and the user made the wrong decision when selecting it. We note that when a user is acquiring an item, one will always incur in a cost which includes the cognitive cost of searching for the item and the real monetary cost eventually paid for the item.

For instance, the designer of a news RS must take into account the complexity of a news item, i.e., its structure, the textual representation, and the time-dependent importance of any news item. But at the same time, the RS designer must understand that even if the user is not paying for reading news, there is always a cognitive cost associated with searching and reading news items. If a selected item is relevant to the user, this cost is dominated by the benefit of having acquired useful information. Whereas if the item is not relevant, the net value of that item for the user, and its recommendation, is negative. In other domains, e.g., cars, or financial investments, the true monetary cost of the items becomes an important element to consider when selecting the most appropriate recommendation approach.

Items with low complexity and value are: news, webpages, books, CDs, and movies. Items with larger complexity and value are: digital cameras, mobile phones, PCs, etc. The most complex items that have been considered are insurance policies, financial investments, travel, and jobs [39].

RSs, according to their core technology, can use a range of properties and features of the items. For example in a movie recommender system, the genre (comedy, thriller, etc.), as well as the director and actors, can be used to describe a movie and to learn how the utility of an item depends on its features. Items can be represented using various information and representation approaches, e.g., in a minimalist way as a single ID code, or in a richer form, as a set of attributes, and even as a concept in an ontological representation of the domain (Chap. 4).

Users Users of an RS, as mentioned above, may have very diverse goals and characteristics. In order to personalize the recommendations and the human-computer interaction, RSs exploit a range of information about the users. This information can be structured in various ways, and again, the selection of what information to model depends on the recommendation technique.

For instance, in collaborative filtering, users are modeled as a simple list containing the ratings provided by the user for certain items. In a demographic RS, sociodemographic attributes such as age, gender, profession, and education, are used. User data is said to constitute the user model [12, 22]. The user model profiles the user, i.e., encodes her preferences and needs. Various user modeling approaches have been used and, in a certain sense, an RS can be viewed as a tool that generates recommendations by building and exploiting user models [10, 11]. Since no personalization is possible without a convenient user model the user model will always play a central role. For instance, in reconsidering a collaborative filtering approach, the user is either profiled directly by its ratings of items or, using these ratings, the system derives a vector of factor values where users differ in how each factor weights in their model (Chaps. 2 and 3).

Users can also be described by their behavior pattern data, for example, site browsing patterns (in a Web-based recommender system) [54], or travel search patterns (in a travel recommender system) [35]. Moreover, user data may include relations between users such as the trust level of these relations between users (Chap. 16). An RS might utilize this information to recommend items to users that were preferred by similar or trusted users.

Transactions We generically refer to a transaction as a recorded interaction between a user and the RS. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using. For instance, a transaction log may contain a reference to the item selected by the user and a description of the context (e.g., the user goal/query) for that particular recommendation. If available, that transaction may also include explicit feedback that the user has provided, such as the rating for the selected item.

In fact, ratings are the most popular form of transaction data that an RS collects. These ratings may be collected explicitly or implicitly. In the explicit collection of ratings, the user is asked to provide an opinion about an item on a rating scale. According to [47], ratings can take on a variety of forms:

- Numerical ratings such as the 1–5 stars provided in the book recommender associated with Amazon.com.
- Ordinal ratings, such as "strongly agree, agree, neutral, disagree, strongly disagree" where the user is asked to select the term that best indicates his or her opinion regarding an item (usually via questionnaire).
- Binary ratings that model choices in which the user is simply asked to decide if a certain item is good or bad.

- Unary ratings can indicate that a user has observed or purchased an item, or otherwise rated the item positively. In such cases, the absence of a rating indicates that we have no information relating the user to the item (perhaps the user purchased the item elsewhere).

Another form of user evaluation consists of tags associated by the user with the items that the system presents. For instance, on Movielens (http://movielens.umn.edu), RS tags represent how MovieLens users feel about a movie, e.g.: "too long," or "acting."

In transactions that collect implicit ratings, the system aims to infer the user's opinion based on the user's actions. For example, if a user enters the keyword "Yoga" at Amazon.com, a long list of books will be provided. In return, the user may click on a certain book on the list in order to receive additional information. At this point, the system may infer that the user is somewhat interested in that book.

In conversational systems, i.e., systems that support an interactive process, the transaction model is more refined. In these systems, user requests alternate with system actions (see Chaps. 10 and 18). That is, the user may request a recommendation and the system may produce a suggestion list. But it can also request additional user preferences to provide the user with better, more refined results. Here, in the transaction model, the system collects the various requests-responses, and may eventually learn to modify its interaction strategy by observing the outcome of the recommendation process [35].

1.4 Recommendation Techniques

In order to implement its core function, identifying useful items for the user, a RS must *predict* that an item is worth recommending. In order to do this, the system must be able to predict the utility of some items, or at least compare the utility of some items, and then decide which items to recommend based on this comparison. The prediction step may not be explicit in the recommendation algorithm but we can still apply this unifying model to describe the general role of an RS. Here, our goal is to provide the reader with a unifying perspective rather than an account of all the different recommendation approaches that will be illustrated in this handbook.

To illustrate the prediction step of an RS, consider for instance, a simple and non-personalized recommendation algorithm that recommends only the most popular songs. The rationale for using this approach is that in the absence of more precise information about the user's preferences, a popular song, i.e., one that is liked (high utility) by many users, will also most-likely appeal to a generic user, or at least with a higher likelihood than another randomly selected song. Hence, the utility of such popular songs is predicted to be reasonably high for this generic user.

This view of the core recommendation computation as the prediction of the utility of an item for a user has been suggested in [2] and recently updated in [44]. Both papers model this degree of utility of the user u for the item i as a (real valued)

function $R(u, i)$, as is normally done in collaborative filtering by considering the ratings of users for items. Then, the fundamental task of a collaborative filtering RS is to predict the value of R over pairs of users and items, or in other words, to compute $\hat{R}(u, i)$, where we denote with $\hat{R}$ the estimation, computed by the RS, of the true function R. Consequently, having computed this prediction for the active user u on a set of items, i.e., $\hat{R}(u, i_1), \ldots, \hat{R}(u, i_N)$, the system will recommend the items $i_{j_1}, \ldots, i_{j_K}$ $(K \leq N)$ with the largest predicted utility. K is typically a small number, that is, much smaller than the cardinality of the item data set or the items on which a user utility prediction can be computed, i.e., RSs "filter" the items that are recommended to users.

As mentioned above, some recommender systems do not fully estimate the utility before making a recommendation, but they may apply some heuristics to hypothesize that an item may be of use to a user. This is typical, for instance, in knowledge-based systems. These utility predictions are computed with specific algorithms (see below) and use various kinds of knowledge about users, items, and the utility function itself (see Sect. 1.3) [17]. For instance, the system may assume that the utility function is Boolean and therefore it will just determine whether an item is or is not useful for the user. Consequently, assuming that there is some available knowledge, or possibly none, about the user who is requesting the recommendation, as well as knowledge about items, and other users who received recommendations, the system will leverage this knowledge with an appropriate algorithm to generate various utility predictions and hence recommendations [17].

It is also important to note that sometimes the user utility for an item is observed to depend on other variables, which we generically call "contextual" [44]. For instance, the utility of an item for a user can be influenced by the domain knowledge of the user (e.g., expert versus beginning users of a digital camera), or can depend on the time when the recommendation is requested. Equally, users may be more interested in items (e.g., restaurant) closer to their current location. Consequently, the recommendations must be adapted to these specific additional details and as a result it becomes increasingly more difficult to correctly estimate what the right recommendations are.

This handbook presents several different types of recommender systems that vary in terms of the addressed domain and the knowledge used, but especially with regard to the recommendation algorithm, i.e., how the prediction of the utility of a recommendation is made, as was mentioned at the beginning of this section. Other differences relate to how the recommendations are finally assembled and presented to the user in response to user requests. These aspects are discussed as well, later in this introduction.

To provide an initial overview of the different types of RSs, we want to quote a taxonomy provided by Burke [17] that has become a classical way of distinguishing between recommender systems and referring to them. Burke [17] distinguishes between six different classes of recommendation approaches:

Content-Based The system learns to recommend items that are similar to the ones that the user liked in the past. The similarity of items is calculated based on the

features associated with the compared items. For example, if a user has positively rated a movie that belongs to the comedy genre, then the system can learn to recommend other movies from this genre [33].

Classic content-based recommendation techniques aim at matching the attributes of the user profile against the attributes of the items. In most cases, the items' attributes are simply keywords that are extracted from the items' descriptions. Semantic indexing techniques represent the item and user profiles using concepts instead of keywords. Chapter 4 presents a comprehensive survey of semantic indexing techniques to overcome the main problems of classical keyword-based systems. The authors presents two main groups of semantic indexing techniques: top-down and bottom-up. Techniques in the former group rely on the integration of external knowledge sources, such as: ontologies, encyclopedic knowledge (such as Wikipedia) and data from the Linked Data cloud, while techniques in the latter group rely on a lightweight semantic representation based on the hypothesis that the meaning of words depends on their usage in large corpora of textual documents. Chapter 4 demonstrates how to utilize semantic approaches to realize a new generation of semantic content-based recommender systems, by providing a description of their main potentials and limitations.

Collaborative Filtering The original and most simple implementation of this approach [24] makes recommendations to the active user based on items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. This is the reason why [48] refers to collaborative filtering as "people-to-people correlation." Collaborative filtering is considered to be the most popular and widely implemented technique in RS.

Chapter 2 presents a comprehensive survey of neighborhood-based methods for collaborative filtering. Neighborhood-based methods focus on relationships between items or, alternatively, between users. An item-item approach models the preference of a user to an item based on ratings of similar items by the same user. Neighborhood-based methods benefit from considerable popularity due to their simplicity, efficiency, and ability to produce accurate and personalized recommendations. Chapter 2 describes the main benefits of such methods, as well as their principal characteristics.

Moreover, Chap. 2 addresses the essential decisions that are required while implementing a neighborhood-based recommender system, and gives practical information on how to make such decisions. Perhaps the decision that has the greatest impact on the rating prediction and computational performance of the recommender system is the choice between a user-based and an item-based method. In typical commercial recommender systems where the number of users exceeds the number of available items, item-based approaches should be preferred since they provide more accurate recommendations, while being more computationally efficient and requiring less frequent updates. On the other hand, user-based methods usually provide more original recommendations, which may lead users to a more satisfying experience [21].

Finally, the problems of sparsity and limited coverage, often observed in large commercial recommender systems, are discussed by exploring two research directions: dimensionality reduction and graph-based techniques. Dimensionality reduction provides a compact representation of users and items that captures their most significant features. An advantage of such an approach is that it allows for obtaining meaningful relations between pairs of users or items, even though these users have rated different items, or these items were rated by different users. On the other hand, graph-based techniques exploit the transitive relations in the data. These techniques also avoid the problems of sparsity and limited coverage by evaluating the relationship between users or items that are not directly connected. However, unlike dimensionality reduction, graph-based methods also preserve some of the "local" relations in the data.

Chapter 3 presents several recent extensions available for building CF recommenders. Specifically, the authors discuss latent factor models, such as matrix factorization (e.g., Singular Value Decomposition (SVD)). These methods transform both items and users to the same latent factor space. The latent space is then used to explain ratings by characterizing both products and users in term of factors automatically inferred from user feedback. The authors elucidate how SVD can handle additional features of the data, including implicit feedback and temporal information. They also describe techniques to address shortcomings of neighborhood techniques by suggesting more rigorous formulations using global optimization techniques. Utilizing such techniques makes it possible to lift the limit on neighborhood size and to address implicit feedback and temporal dynamics. The resulting accuracy is close to that of matrix factorization models, while offering a number of practical advantages.

Demographic This type of system recommends items based on the demographic profile of the user [13]. The assumption is that different recommendations should be generated for different demographic niches. Many websites adopt simple and effective personalization solutions based on demographics. For example, users are dispatched to particular websites based on their language or country. Or, suggestions may be customized according to the age of the user. While these approaches have been quite popular in the marketing literature, there has been relatively little proper RS research on demographic systems.

Knowledge-Based Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users' needs and preferences and, ultimately, how the item is useful for the user. Notable knowledge-based recommender systems are case-based [15, 19, 45]. In these systems, a similarity function estimates how much the user's needs (problem description) match the recommendations (solutions of the problem). Here, the similarity score can be directly interpreted as the utility of the recommendation for the user. Knowledge-based systems tend to work better than others at the beginning of their deployment but if they are not equipped with learning components, they may be surpassed by other shallow methods that can exploit the logs of the human/computer interaction (as in CF).

Constraint-based systems are another type of knowledge-based RS (Chap. 5). In terms of used knowledge, both systems are similar: user requirements are collected, repairs for inconsistent requirements are automatically proposed in situations where no solutions could be found, and recommendation results are explained. The major difference lies in the way solutions are calculated. Case-based recommenders determine recommendations on the basis of similarity metrics whereas constraint-based recommenders predominantly exploit predefined knowledge bases that contain explicit rules about how to relate customer requirements with item features.

Chapter 5 reviews constraint-based recommendation approaches and provide an overview of technologies for the development of knowledge bases for constraint-based recommenders since appropriate tool support can be crucial in practical settings. The authors show that constraint-based methods are particularly well suited for recommending complex products such as financial services or electronic consumer goods.

Moreover, Chap. 5 presents possible forms of user interaction that are supported by constraint-based recommender applications, report scenarios in which constraint-based recommenders have been successfully applied, and review different technical solution approaches.

Community-Based This type of system recommends items based on the preferences of the user's friends. This technique follows the epigram, "Tell me who your friends are, and I will tell you who you are" [4, 9]. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals [52]. This observation, combined with the growing popularity of open social networks, is generating a rising interest in community-based systems or, as they are usually referred, social recommender systems [23]. This type of RS models and acquires information about the social relations of the users and the preferences of the user's friends. The recommendation is based on ratings that were provided by the user's friends. In fact these RSs are following the rise of social-networks and enable a simple and comprehensive acquisition of data related to the social relations of the users.

Hybrid Recommender Systems These RSs are based on the combination of the above mentioned techniques. A hybrid system combining techniques A and B tries to use the advantages of A to fix the disadvantages of B. For instance, CF methods suffer from new-item problems, or, that they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available. Given two (or more) basic RSs techniques, several ways have been proposed for combining them to create a new hybrid system (see [17] for the precise descriptions).

As we have already mentioned, the context of the user when he or she is seeking a recommendation can be used to better personalize the output of the system. For example, in a temporal context, vacation recommendations in winter should be very different from those provided in summer [8]. Or a restaurant recommendation for a Saturday evening with one's friends should be different from that suggested for a workday lunch with co-workers.

Chapter 6 reviews the topic of context-aware recommender systems (CARS). It presents the general notion of context and how it can be modeled in RSs. As it discusses the possibilities of combining several context-aware recommendation techniques into a single unified approach, the authors also provide a case study of one such combined approach.

Three popular different algorithmic paradigms for incorporating contextual information into the recommendation process are discussed: reduction-based (prefiltering), contextual post filtering, and context modeling. In reduction-based (prefiltering) methods, only the information that matches the current usage context, e.g., the ratings for items evaluated in the same context, are used to compute the recommendations. In contextual post filtering, the recommendation algorithm ignores the context information. The output of the algorithm is filtered/adjusted to include only the recommendations that are relevant in the target context. In the contextual modeling, the more sophisticated of the three approaches, context data is explicitly used in the prediction model.

Recommendation tasks can be solved with the help of techniques that were developed in the field of Data Mining. Chapter 7, presents an overview of the main Data Mining techniques used in the context of Recommender Systems and presents cases where these techniques have been successfully applied. In particular, it discusses the following techniques: preprocessing techniques such as sampling or dimensionality reduction; classification techniques, such as Bayesian Networks, Decision Trees and Support Vector Machines; clustering techniques such as k-means; and finally association rules.

1.5 Recommender Systems Evaluation

Recommender systems research is being conducted with a strong emphasis on practice and commercial applications. One very important issue related to the practical side of RS deployment is the necessity of evaluating the quality and value of the systems. Evaluation is required at different stages of the system's life cycle and for various purposes [1, 26]. At design time, evaluation is required to verify the selection of the appropriate recommender approach. In the design phase, evaluation should be implemented off-line and the recommendation algorithms, i.e., their computed recommendations, are compared with the stored user interactions. An off-line evaluation consists of running several algorithms on the same datasets of user interactions (e.g., ratings) and comparing their performances. This type of evaluation is usually conducted on existing public benchmark data if appropriate data is available, or, otherwise, on collected data. The design of the off-line experiments should follow known experiment design practices [6] in order to ensure reliable results. Off-line experiments can measure the quality of the chosen algorithm in fulfilling its recommendation task. However, such evaluation cannot provide any insight about the user satisfaction, acceptance or experience with the system. The algorithms might be very accurate in solving the core recommendation

problem, i.e., predicting user ratings, but for some other reason the system may not be accepted by users, for example, because the performance of the system was not as expected.

Therefore, a user-centric evaluation is also required. It can be performed online after the system has been launched, or as a focused user study. During on-line evaluation, real users interact with the system without being aware of the full nature of the experiment running in the background. It is possible to run various versions of the algorithms on different groups of users for comparison and analysis of the system logs in order to enhance system performance. In addition, most of the algorithms include parameters, such as weight thresholds, the number of neighbors, etc., requiring constant adjustment and calibration.

Focused user studies are conducted when the on-line evaluation is not feasible or too risky. In this type of evaluation, a controlled experiment is planned where a small group of users are asked to perform different tasks with various versions of the system. It is then possible to analyze the user's performance and to distribute questionnaires so that users may report on their experience. In such experiments, it is possible to collect both quantitative and qualitative information about the systems.

In recent years there has been an increased interest in user-centric evaluation procedures and metric for recommender systems. Researchers realized that recommender systems' goals extend beyond the accuracy of the algorithms [30] as tools to provide a helpful and enjoyable, personalized experience that leads to user retention and satisfaction. This approach broadened the range of evaluated aspects of an RS to include aspects such as the form of preference elicitation, the presentation of the recommended results (e.g., one top item, top N items, or predicted ratings), and finally, the evaluation of the explanations provided to the users. Explanations may serve a few goals: the most popular is the justification of results, i.e., explaining to the user why the system decided to recommend a specific item. Other goals may include increasing trust in the system, persuading the user to purchase the recommended item, and helping a user with their decision making. When designing the evaluation of the recommendation explanation, it is important to identify the goal of the explanation and adjust a suitable metric for measuring it.

Chapter 8 details the three previously mentioned types of experiments that can be conducted in order to evaluate recommender systems, namely, off-line, on-line and user studies. It presents their advantages and disadvantages, and defines guidelines for choosing the methods for evaluating them by considering the properties that are to be evaluated. Unlike existing discussions of evaluation in the literature that usually focuses mainly on the accuracy of an algorithm's prediction [26] and related measures, this chapter is unique in its approach to the evaluation discussion since it focuses on property-directed evaluation. It provides a large set of properties (other than accuracy) that are relevant to the system's success. For each of the properties, the appropriate type of experiment and relevant measures are suggested. Among the list of properties are: coverage, cold start, confidence, trust, novelty, risk, and serendipity. The chapter describes the difficulties and pitfalls of each of the properties and guidelines for the selection of the suitable evaluation type and properties for a given recommendation task ad system.

Chapter 9 highlights the importance of user-centric evaluation. It provides detailed and practical guidelines of how to conduct user-centric experiments in order to evaluate the user's experience with the system. The chapter first presents a theoretical user-centric evaluation framework [29] that maps aspects of recommender systems and their interaction with the users that should be evaluated. Then, it provides practical guidelines for students and researchers for conducting user experiments. It presents tips for stating hypotheses, recruiting participants, design of the experiments and statistical analysis of the results. Finally, the chapter provides many examples of actual systems' evaluations from the relevant literature.

Chapter 10 tackles an additional aspect of the user-centric approach to evaluation and highlights an important aspect of RS: explanations of the recommendation results to the users. It examines the reasons that make an evaluation "good" and the effects that explanations might have on RS acceptance. The chapter first explains the interaction between the recommender system and the explanation in terms of preference elicitation methods and the presentation of results, as well as the recommendation algorithm. Then, explanation styles are described, along with examples of explanation in existing systems. The goals of explanations are listed from which metrics that measure the success of explanations in achieving these goals are described. The chapter concludes with challenges related to explanations. This includes the context in which explanations should be shown, and a major challenge in evaluating the interaction between acceptance of recommendation and explanations, as well as how to assure that explanations are indeed helpful and do not lead users to make poor decisions.

1.6 Recommender Systems Applications

Recommender systems research, aside from its theoretical contribution, is generally aimed at practically improving industrial RSs and involves research about various practical aspects that apply to the implementation of the systems. Indeed, an RS is an example of large scale usage of machine learning and data mining algorithms in commercial practice [3]. The common interest in the field, both from the research community and from the industry has leveraged the availability of data for research on one hand, and the evolvement of enhanced algorithms on the other hand. Practical related research in RSs examines aspects that are relevant to different stages in the life cycle of an RS, namely, the design of the system, its implementation, evaluation, maintenance and enhancement during system operation. The Netflix Prize announced in 2006, described in Chap. 11, was an important event for the recommender systems research community and industry, and their mutual interaction. It highlighted the importance of the recommendation of items to users and accelerated the development of many new data mining recommendation techniques. Even though the Netflix Prize initiated a lot of research activities, the prize was a simplification of the full recommendation problem. It consisted of predicting user's ratings while optimizing the Root Mean Square Error (RMSE)

between the predicted and actual ratings. Chapter 11 describes lessons learned from the prize awarded in 2009 and provides insights about industrial setting of RSs using the Netflix system as a case study for real-world recommender systems. In addition, the chapter provides the industry perspective about RSs implementation issues that deserve attention while developing a real-world RS. It describes the data centric approach used at Netflix for selecting the best model for a problem in order to provide an optimized personalized experience to the user. In addition, the chapter highlights the need for a suitable scalable system architecture that can support the development and evaluation process of innovative algorithms and deliver recommendations based on large volumes of available data.

The first factor to consider while designing an RS is the application's domain as it has a major effect on the algorithmic approach that should be taken. Montaner et al. [39] provide a taxonomy of RSs and classify existing RS applications to specific application domains. Based on these specific application domains, we define more general classes of domains for the most common recommender systems applications:

- Entertainment—recommendations for movies, music, games, and IPTV.
- Content—personalized newspapers, recommendation for documents, recommendations of webpages, e-learning applications, and e-mail filters.
- E-commerce—recommendations of products to buy such as books, cameras, PCs etc. for consumers.
- Services—recommendations of travel services, recommendation of experts for consultation, recommendation of houses to rent, or matchmaking services.
- Social—recommendation of people in social networks, and recommendations of content social media content such as tweets, Facebook feeds, LinkedIn updates, and others.

As recommender systems become more popular, interest is roused in the potential advantages of new and diverse applications, such as recommending insurance riders, or recommending questions for question-answering systems. As the above list cannot cover all the application domains that are now being addressed by RS techniques: it gives only an initial description of the various types of application domains.

The developer of an RS for a certain application domain should understand the specific facets of the domain, its requirements, application challenges and limitations. Only after analyzing these factors can one be able to select the optimal recommender algorithm and to design an effective human-computer interaction. In the current version of the handbook, some of the chapters in this section describe applications of recommender systems in specific domains. Each of these chapters describes the requirements of an RS for a specific domain, its precise challenges and the suitable technologies and algorithms for addressing them.

One detailed example of an RS designed for a specific domain is described in Chap. 12, which deals with recommender systems for technology-enhanced learning (TEL). TEL, which generally covers technologies that support all forms of teaching and learning activities, aims at designing, developing and testing new methods and

technologies to enhance learning practices of both individuals and organizations. As the digital way of education and learning is becoming more popular, the need to integrate the personalization of content into the learning process and the available data for assessing the quality of the algorithms has created opportunities for the rise of the popularity of TEL RSs. Since TEL may benefit greatly from integrating recommender systems technology to personalize the learning process and adjust it to the user's former knowledge, abilities and preferences [55], there is a significant increase in RSs applied to TEL. The chapter presents an extensive survey of RSs for TEL covering 82 systems from 35 countries categorizing them using a classification framework consisting of three main categories, namely: Supported Tasks, Approach, and Operation. An analysis of the 82 systems using the framework resulted in seven clusters of TEL RSs, where each cluster represents a unique form of contribution to the field. The chapter aims at providing an overview about the TEL RS field in order to standardize evaluation settings and measures, as well as providing guidelines for the application of RS technology to TEL.

Yet another popular domain for recommendation is the recommendation of music, presented in Chap. 13. Unique features of music items that pose various challenges for recommendations should be considered when designing and evaluating RSs for music. Such challenges include, for example, the short time that it takes a user to gain an opinion about a recommended item, as compared to a movie or a book, or the fact that the same item can be recommended many times. In addition, music can be recommended as a single item, a playlist, and abstracted by genre, performer, or band. Music RSs, as opposed to many other domains, rely heavily on content-based recommendation which implies specific challenges to the domain [37].

Some new application domains for recommendation evolved with the emergence of new technologies that became very popular. One example, detailed in Chap. 14, is the evolution of mobile technology that accelerated the development of specific RSs for mobile devices utilizing the special capabilities of the devices (e.g., the GPS). Chapter 14 reviews the main components of a location based mobile recommender system. While there are various application domains of mobile context aware RSs (as described in Chap. 6) that may benefit from mobile sensors in order to enrich their users' profiles, Chap. 14 highlights mobile location-based recommendations. Such RS applications recommend places and venues based on the user's location and history of behaviors and preferences. The chapter describes the algorithms that have been applied to recommending venues, and the evaluation procedures for assessing the quality of the recommendations. Looking into the future, the authors suggest additional location based applications such as recommendations for cab drivers on pick-up locations, or recommending where to locate a retail store.

Another example of new RSs that emerged with new technologies are recommender systems related to the social web, and specifically those that target the social media domain. With the rise of social networks (e.g., Facebook, LinkedIn, Tweeter, Flickr, and others), users are overloaded with information, activities and interactions. Social recommender systems are RSs that aim at assisting the user in identifying relevant content (e.g., tweets, feeds or images), and engage only in

relevant activities and interactions (e.g., discussions, or comments). Apart from the RSs that have been developed to be dedicated to social media, recommender systems of other domains can benefit from the new types of data that social media introduces about users to enhance the quality of standard RSs [23]. The term Social RS covers many types of RSs that are relevant to the social media platform in which Chap. 15 describes two main types: recommendations of social media content and recommendations of people. For recommendations of social content, the chapter reviews various social content media domains, and provides a detailed case study and insights learned from a recommender system operated in the enterprise which suggests mixed social media items. The chapter lists three different types of people recommendations, namely, the recommendation of familiar people (e.g., classmates, family members) that are not connected in the network; recommendations of interesting people (to connect with, or follow), and recommendations of strangers (to date, to hire, or for various other purposes). It explains the complexity of people-recommendation and lists key topics that should be considered and should be further investigated. The list includes: the need for explanation, privacy concerns, social relationships, trust and reputation, as well as the need to define special evaluation measures.

Chapter 16 highlights a special form of social recommendations, the reciprocal people-to-people recommendations that require the two parties involved in a recommendation to be satisfied. Some examples are: dating recommendations, Human Resource recommendations (recommendations of employees and employers), and recommending groupings of students for learning groups. Besides the unique reciprocal manner of satisfaction that is required from both parties, the chapter highlights other differences between traditional and reciprocal recommendations, including the users' willingness to provide explicit feedback, the fact that users are usually engaged with the system for a long term, and the requirement not to overload users with recommendations.

Within the chapter, the authors provide an overview regarding existing reciprocal recommender systems and demonstrate how the special requirements of reciprocity are considered in system design. A detailed case study, specifically in online dating recommendation, is presented and includes the recommendation hybrid content-collaborative algorithm and its evaluation. One interesting insight shown is that implicit feedback is more effective than explicit for these types of systems.

The social Web is also exploited by modern search engines that rely on recommendation techniques to address Web search challenges and to implement advanced search features. Specifically, various engines attempt to apply some form of personalization and collaboration by generating results to a user query that are not only relevant to the query terms but are also tailored to the user's search history, reputation, and preferences as inferred from the user's own previous activities on the social Web.

Chapter 17 discusses the research goals of Information Retrieval (IR) and personalized Web search from the RS perspective. The authors illustrate how techniques that originated in recent RS research may be applied to address search engine challenges. This chapter focuses on two promising ideas for search engine

improvement: personalization and collaboration. It describes a number of different approaches to personalizing Web searches by exploiting user preferences and context information to affect search results. In addition, the chapter discusses recent work in the area of collaborative information retrieval, which attempts to take advantage of the potential for cooperation between friends, colleagues or users with similar needs in implementing a variety of information-seeking tasks. This new line of research, termed social search, benefits from the social medium property of the Web in providing search results that are affected by the experience and preferences of similar users. The authors foresee a convergence of recommender systems and search engines, where a search engine will provide a unique platform for modern recommendation technologies. The authors believe that integrating these sources in search engine algorithms, along with a proactive manner of search experience that strives to understand the users' needs, would result in highly satisfied users that are able to receive the right information at the right time. Another trend that is affecting search engines is the rise of search activities through mobile devices. This introduces new constraints to search and discovery interfaces, but also brings about opportunities for innovations using the mobile sensors that allow enhanced personalization.

1.7 Recommender Systems and Human Computer Interaction

As we have illustrated in the previous sections, researchers have been chiefly concerned with designing a range of technical solutions and leveraging various sources of knowledge to achieve better predictions about what is liked and how much it is liked by the target user. The underlying assumption behind this research activity is that simply presenting these correct recommendations, or the best options, is not sufficient. In other words, the recommendations should speak for themselves, and the user should definitely accept the recommendations if they are correct. This is clearly an overly simplified account of the recommendation problem and the delivery of recommendations is not so straightforward.

In practice, users need recommendations because they do not have enough knowledge to make an autonomous decision. Consequently, it may not be easy for them to evaluate the proposed recommendation. Hence, various researchers have tried to understand the factors that lead to the acceptance of a recommendation by a given user [5, 20, 25, 38, 50, 53].

Swearingen and Sinha [53] were among the first to point out that the effectiveness of an RS is dependent on factors that go beyond the quality of the prediction algorithm. In fact, the recommender must also convince users to try (or read, buy, listen, watch, etc.) the recommended items. This, of course, depends on the individual characteristics of the selected items, and therefore on the recommendation algorithm. The process also depends, however, on the particular human/computer

interaction supported by the system when the items are presented, compared, and explained. Swearingen and Sinha [53] found that from a user's perspective, an effective recommender system must inspire trust in the system and it must have a system logic that is at least somewhat transparent. Additionally, the authors note that it should point users towards new, not-yet-experienced items, and should provide details about recommended items, including pictures and community ratings, and finally, it should present ways to refine recommendations.

Swearingen and Sinha [53] and other similarly oriented researchers do not diminish the importance of the recommendation algorithm, but claim that its effectiveness should not be evaluated only in terms of the accuracy of the prediction, i.e., with standard and popular IR metrics, such as Mean Absolute Error (MAE), precision, or Normalized Discounted Cumulative Gain (NDCG) (see Chap. 8). Other dimensions should be measured that relate to the acceptance of the recommender system and its recommendations. These ideas have been remarkably well presented and discussed also by McNee et al. [38]. In that work, the authors propose user-centric directions for evaluating recommender systems, including: the similarity of recommendation lists, recommendation serendipity, and the importance of user needs and expectations in a recommender.

Recommender systems collect tremendous amounts of user data that are necessary for the recommendation purposes. However, the availability of this data may result in this data being used in a way that violates the end-user's expectations of privacy, especially if it is accessed by untrusted parties or misused by malicious agents. Chapter 19 presents the latest in privacy enhanced recommendations. The authors analyze the risks to user privacy imposed by recommender systems, survey the existing solutions, and discuss the privacy implications for the users of the recommenders. In particular, the authors describe several architectures that preserve user privacy by using various decentralized solutions that eliminate a single repository of user modeling data, which would otherwise be the target for malicious attacks on the recommender. In addition, the authors present algorithmic solutions, which either perturb the original user modeling data or apply formal encryption methods. These assure that, even if accessed by an untrusted party, only modified or encrypted user data would be exposed rather than the original data. Lastly, the policy driven solutions are described. These solutions address directives and legislation initiatives that limit the storage, transfer, and exploitation of personal user data.

An essential goal of recommender systems is to help users make better choices [18]. Thus, it is important to understand how people make choices and how the human decision making process can be supported. Chapter 18 begins with a compact overview of the psychology of everyday choice and decision making that is based on a large literature of psychological research and formulated so as to be relevant and accessible to recommender systems research community. The authors explain how recommender systems can be viewed as one of many available tools for facilitating choice. Then, the authors provide a high-level overview of strategies for helping people make better choices, indicating how recommender systems fit into the greater picture of choice. In addition, the main functionalities of RSs are presented: eliciting information to construct preference models, narrowing down a large set of options,

helping users choose among a small set of recommended options, and helping users to explore large spaces of options. The authors show how an understanding of human decision making can illuminate research and practice concerning these processes.

As discussed in previous sections, recommender systems often utilize sophisticated algorithms to make recommendations. However, the RS cannot assume the advice provided by a system will always be accepted by its users. Whether a recommendation is seen as credible advice and actually taken into account not only depends on users' perceptions of the recommendation but also of the system as an advice-giver.

In Chap. 20 the authors stress that a recommendation is seen as credible advice and is actually taken into account not only because of the user's perceptions of the recommendation but also due to the fundamental role of the system which is perceived as an advice-giver. Indeed, the literature about persuasion suggests that people are likely to accept recommendations from credible sources and we therefore conclude that the credibility of the RS is vital to increasing the likelihood of recommendation acceptance. Hence, the authors discuss how the credibility of RSs can be enhanced, providing a synopsis of credibility-related research.

Chapter 20 reviews the existing literature on source factors in the context of human-human, human-technology, and human-recommender system interactions. It also discusses system credibility evaluation in light of the increasing popularity of social technology. Source characteristics which have been studied in the context of human and technology interaction, and particularly, in the recommender systems realm are discussed as well. Finally, Chap. 20 concludes that many social cues that have been identified as influential in other contexts have yet to be implemented and tested with respect to recommender systems.

Personality accounts for the most important way in which individuals differ in their enduring emotional, interpersonal, experiential, attitudinal and motivational styles. Studies have shown that personality was especially useful at tackling the issues of the cold start problem and diverse recommendations.

Chapter 21 discusses how personality relates to user preferences and how to use personality in recommender systems. The authors present the Five Factor Model (FFM) of personality. This model appears suitable for usage in recommender systems as it can be easily quantified in terms of features corresponding to the main factors. The acquisition of the personality factors for an observed user can be made explicitly through questionnaires or implicitly using machine learning approaches on modalities like social media streams or mobile phone call logs.

1.8 Advanced Topics

It is clear from the previous pages that RS research is evolving in numerous and diverse directions, and new topics are emerging or becoming more important subjects of investigation. The reader is also encouraged to refer to the proceedings

of the last editions of the ACM RecSys conferences and other excellent review papers for additional material [13, 30, 34, 40, 44]. In this handbook, we cover some of these topics. Indeed, some have already been presented, such as: context-aware recommendations (Chap. 6), social-based recommendations (Chap. 15), and reciprocal recommendations (Chap. 16). Other important topics are covered in the last section of this handbook and will now briefly introduce these chapters.

Chapter 22 deals with situations in which the system should recommend information or items that are relevant to a group of users rather than to an individual. For instance, a RS may select television programs for a group to view or a sequence of songs to listen to, based on models of all the group members. Recommending to groups is clearly more complicated than recommending to individuals. Assuming that we know precisely what is good for individual users, the issue is how to combine individual user models. In this chapter, the authors discuss how group recommendation works, what its problems are, and what advances have been made so far.

Chapter 23 discusses the ubiquitous issue of aggregating preferences, criteria or similarities. Normally such aggregation is done by using either the arithmetic mean or maximum/minimum functions. But many other aggregation functions which could deliver flexibility and adaptability, and ultimately more relevant recommendations, are often overlooked. In this chapter the authors review the basics of aggregation functions and their properties and present the most important families, including generalized means, Choquet and Sugeno integrals, ordered weighted averaging, triangular norms and conorms, as well as bipolar aggregation functions. Such functions can model various interactions between the inputs, including conjunctive, disjunctive and mixed behavior.

In Chap. 24, the authors focus on another fundamental problem of RSs: the need to actively look for new data during the operational life of the recommender. This issue is normally neglected on the assumption that there is not much space for controlling the data (e.g., ratings) that the system can collect, since these decisions are taken by the users when visiting the system. Actually, the RS provokes the users with its recommendations and many systems actually explicitly ask for user preferences during the recommendation process. Hence, by tuning the process, users can be pushed to provide a range of different information. Specifically they can be requested to rate particular items since the knowledge of the user's opinions about these items could be estimated as particularly beneficial to the system performance. For instance, the system may be able to provide more diverse recommendations with this additional information, or may improve its prediction accuracy. At this point active learning comes in; it can augment RSs, helping users to become more self-aware of their own likes/dislikes, and lead to more meaningful and useful questions. At the same time, active learning can provide new information to the system that can be analyzed for subsequent recommendations. Hence, applying active learning to RSs enables personalization of the recommending process [36]. This is accomplished by allowing the system to actively influence the items that the user is exposed to (e.g., the items displayed to the user during sign-up or during regular use), as well as by enabling the user to explore his or her interests freely.

Chapter 25 introduces another emerging topic: multi-criteria recommender systems. In the majority of RSs, the utility associated with an item is usually considered a single criterion value, or an overall evaluation or rating of an item by a user. But recently, this assumption has been judged as limited because the suitability of the recommended item for a particular user may depend on several aspects that the user can take into consideration when making his or her choice. The incorporation of multiple criteria that can affect the user's opinions may lead to more effective and accurate recommendations.

Chapter 25 provides an overview of multi-criteria RSs. First, it defines the recommendation problem as a multi-criteria decision-making problem and reviews methods and techniques that can support the implementation of multi-criteria recommenders. Then, it focuses on the category of multi-criteria rating recommender techniques that provide recommendations by modeling the user's utility for an item as a vector of ratings along several criteria. A review of current algorithms that use multi-criteria ratings for calculating the rating prediction and generating recommendations is provided. The chapter concludes with a discussion on open issues and future challenges for these recommenders.

Accurately predicting that the user surely likes a small number of items and suggesting them repeatedly it is not likely to provide a quality service to the user. Likewise, if the suggested items are all quite similar, the system's suggestions will seem repetitive and it may fail to identify peculiar items. Such peculiar items, though they are at the boundary of the user's spectrum of preference, may be unexpected and therefore perceived as more informative or highly desirable to the user.

These topics are discussed in Chap. 26 that is dedicated to novelty and diversity in recommender systems. After having motivated the need for novel and diverse recommendations, the chapter precisely defines these two naturally vague concepts and their relationships. While novelty is a property of the individual recommendations, diversity can only be measured by referring to sets of recommendations, either for a single user or for the full set of users of a recommender system. The quantitative measure of these two properties occupies a substantial amount of the chapter in that this is an important preliminary step prior to focusing on techniques that will improve the system's performance in this respect. This issue is then discussed by presenting a range of techniques aimed at enhancing a recommendation list by editing it with the ultimate goal of increasing the diversification of the recommendations and their novelty. Finally the chapter presents a unifying model that can describe the range of metrics presented thus far. A critical element in this unifying model is that novelty is relative to a context of experience, showing the unexpected relationship between this line of research and that on context-aware recommender systems (see Chap. 6).

As we have referred to already in this introduction, recommender system applications are normally restricted to a particular type of items (movies, CDs, books, etc.). Even though an ecommerce player sells several different types of products (e.g., Amazon), its recommender system treats each product typology independently from the others. This means that if a user has expressed his preferences for a certain category of products (e.g., books), for instance in the form of ratings, this

information is ignored when the system computes recommendations for items in another category (e.g., movies). But, it is natural to expect that preferences in a domain are somewhat correlated to preferences in another domain. So, for instance, if two users have bought very similar sets of books, it is likely that some of the movies bought by one of the users may be good recommendations for the other user.

The idea of relating recommendations in different domains by exploiting user data collected in one domain to produce recommendations in another, is at the base of the research on cross-domain recommender systems. These systems and their underlying techniques are illustrated in Chap. 27. It is shown here how leveraging all the user's preferences, collected in several domain specific systems, may be beneficial for generating more comprehensive user models and better recommendations. Cross-domain recommender systems can in fact mitigate the cold-start and sparsity problems in a target domain where not much user data has been collected. Moreover, they can enable cross-selling recommendations, such as building more complex recommendations where items from multiple domains are suggested together (e.g, a movie and a book on a recommended singer). In this chapter, the authors formally define the cross-domain recommendation problem, and try to provide a unifying perspective by merging ideas and approaches which arise in distinct disciplines. The chapter provides an analytical categorization of prior work, and identifies open issues for future research.

The last chapter of this handbook (Chap. 28) surveys articles which deal with security issues. This topic has become a major issue in the past few years. The chapter analyzes algorithms designed to generate more robust recommendations, i.e., recommendations that are harder for malicious users to influence. In fact, collaborative recommender systems are dependent on the goodwill of their users, in that there is an implicit assumption that users will interact with the system with the aim of getting good recommendations for themselves while providing useful data for their neighbors. However, users will have a range of purposes in interacting with RSs and in some cases, these purposes may be counter those of the system owner or those of the majority of the user population. Namely, these users may want to damage the website which hosts the recommender, or to influence the recommendations provided to visitors, e.g., to score some items better or worse rather than to arrive at a fair evaluation.

In this chapter, the authors provide a model of efficient attacks, i.e, attacks that can, at a relatively low cost, produce a large impact on system output. Since these attacks may very well be launched against a site, it makes sense to detect them so that countermeasures can be taken as soon as possible. At the same time, researchers have studied a number of algorithms that are intended to robustly withstand attacks and which have lower impact curves relative to efficient attacks. These approaches are also surveyed in this chapter. With the combination of these techniques, researchers have sought not to eliminate attacks, but rather to control their impact to the point in which they are no longer cost-effective.

1.9 Challenges

The list of newly emerging and challenging RS research topics is not limited to those described in the chapters that we have summarized above. Moreover, covering all of them is not within the scope of this introduction. The reader is referred to the final discussion sections that are included in almost all of the chapters published in this handbook for other crucial problems.

Below, we briefly introduce additional challenging topics that we consider important for the development of the research on RSs.

1.9.1 Preference Acquisition and Profiling

A number of open issues are related to the critical stage of acquiring information about the user preferences and generating usable profiles of the users.

It is clear that in many real-world situations, implicit feedback is much more readily available and requires no extra effort on the user's side. For instance, on a web page it is easy to log the users visiting a URL, or clicking on an ad. The system can treat these actions as a form of positive feedback to the displayed items. It makes sense that information about such previous actions contains highly relevant information for predicting future actions. For that reason many recent approaches focus on the use of the more reliable and readily available implicit feedback (see the final discussion in Chap. 11 that refers to this issue). In cases where we have implicit feedback, the recommendation problem becomes the prediction of the probability that a user will interact with a given item. But, standard recommendation formulation in such a setting is not applicable: there is no negative feedback, all the available data is either positive or missing. The missing data includes both items that the user explicitly chose to ignore because they were not appealing and items that would have been perfect recommendations but were never presented to the user. This issue is discussed in Chap. 11 and effective solutions for tackling it must be still developed further.

Notwithstanding the pros of implicit feedback, this data cannot completely substitute the usage of explicitly user-made evaluations, at last for some of the items. In order to make the acquisition of this information more effective, as is illustrated in Chap. 24, a number of techniques for actively collecting preference data from users, especially in the cold start phase, but also in the full life of a system, have been illustrated. It is however still challenging to select, among the various active learning techniques, the one that is more appropriate to a system when dealing with a particular user in a given stage of the system evolution and for a particular goal (e.g., accuracy vs. coverage). More adaptive solutions which can blend, run-time with several elementary approaches, should be investigated. Moreover, their

practical application depends also on their computational cost, which could become prohibitive when multiple strategies must be estimated in real time to find out the best one for that specific scenario.

Hence, in addition to better active learning techniques, simplifying the cognitive cost of preference acquisition is of primary importance. For a recommender system to achieve good recommendation performance, users typically need to provide the system with a certain amount of feedback about their preferences (e.g., in the form of item ratings). This can be an issue in single-rating recommender systems and even more for multi-criteria rating systems that require a more significant level of user involvement, as each user needs to rate an item based on multiple criteria. Therefore, it is important to measure the costs and benefits of adopting alternative rating approaches and scales, and find an optimal solution to meet the needs of both the users and the system designers. For instance, preference disaggregation methods could support the implicit formulation of a preference Multi-Criteria Recommender Systems model (as indicated in Chap. 25).

Another direction of research for easing the preference acquisition process consists of the exploitation of personality, mood and emotions. This is becoming a popular topic, especially because it is clear that more and more techniques will be developed in order to automatically acquire such information. In Chap. 21, the authors stress the challenge of acquiring personality information in a nonintrusive fashion. Nowadays, only the longest questionnaires, which consist of around one hundred questions, can provide an accurate evaluation of the user's personality. Hence, non-intrusive approaches are necessary and the research in this area is just starting. Mining user activity for extracting personality information is an option, but also the fast penetration of portable devices that are life-logging the user's activity can offer a promising platform that is worth exploring.

Another line of research aimed at tackling the cold start problem and reducing the user model elicitation effort is cross-domain recommender systems. These techniques (See Chap. 27) could be used as an alternative path to user preferences' elicitation tools as they are able to build detailed user profiles without the need to collect explicit user assessment of the target domain items. Finally, we want to mention the issue of integrating long-term and short-term user preferences into the process of building a user profile and delivering a recommendation list. Recommender systems may be divided into two classes: those that build a long-term profile, generated by aggregating all the user transaction data collected by the system (e.g., collaborative filtering) and those that are more focused on capturing the ephemeral preferences of the user (e.g., knowledge-based approaches), such as case-based. Obviously both aspects are important and either the precise user task or the availability of items may come into consideration in resolving the preference integration problem. In fact, new research is required to build hybrid models that can correctly decide to what extent to drift toward the contingent user's preferences when there is enough evidence suggesting that the user's short-term preferences are departing from the long-term ones.

1.9.2 Interaction

A major challenge that RS research is now facing is clearly discussed in Chap. 9, in that we still need to broaden the scope of research to the system aspects of a recommender system. This means that aside from the algorithms, which are used to compute the recommendations, the mechanism through which users provide their input and the means by which they receive the systems output, play a significant role and can play an even larger role in determining the success or failure of a recommender system. We still need to better understand the general qualities of alternative solutions to preference elicitation, as we mentioned previously, recommendation presentation and to develop personalized solutions for these phases of the interaction with the system.

It must be observed that while interacting with a recommender system, users make various types of decisions. The most important one is surely selecting an item from the recommendation list. But, before making the final decision, users often have to decide how to explore the information space and what information they must provide to the system. For instance, they could have to select a specific feature (e.g., a camera's size or zoom factor) as search or critiquing criteria, or to select a repair proposal for inconsistent user preferences when interacting with a knowledge-based recommender. Moreover, users often do not know or do not reflect on their preferences beforehand, and the system-supported interaction and visualization contribute to the user construction of their preferences within a specific recommendation scenario. As it has been illustrated in Chap. 18, there are several challenges with the full support of user decision making in a recommender system. Our understanding of the situational context generated by the system and its effect on item selection processes is still incomplete and we need to better connect RS research to psychology and decision making disciplines. While it is clear that RS helps to make decisions, there is still the need for further research that takes theories from decision psychology and cognitive psychology into account when explaining users' preference construction and decision making process in the context of recommender systems.

Considering the user interaction with the recommender system, the topic of explaining the system recommendations still poses a number of interesting and open issues (see Chap. 10). For instance, it is still not completely clear whether explanations bring more overall benefits than risks. In Chap. 10 it is shown that explanations are part of a cyclical process: the explanations affect the acceptance of particular recommendations, the users' mental model of the system, and in turn, this affects the ways users interact with the explanations. But, whether the users are influenced in such a way that their choices are improved is not clear, and explanations may even increase the information overload that the recommendations are supposed to tame. Moreover, while some research has been conducted on explaining recommendations to individual users, explaining recommendations for a group is a much more novel subject. For instance, one might think that accurate predictions of individual satisfaction can also be used to improve the recommender's

transparency: showing how satisfied other group members are could improve users' understanding of the recommendation process and perhaps make it easier to accept items they do not like. However, users need for privacy is likely to conflict with their need for transparency and showing the preferences of other users may move the group discussion on the preferences rather than on the recommended items. We definitely need more research on these topics.

In a discussion about the interaction with recommender systems, we cannot forget the issue of the assessment of the value of the recommendations, which is not only related to what extent the recommended items are liked by the user. For instance, the time value of recommendations, which is partially discussed in the chapter on context-aware recommenders (Chap. 6), refers to the fact that a given set of recommendations may not be applicable forever but there could be a time interval when these items can be recommended. This is clear, for instance, when it comes to news items: people want to be informed about the most recent events and news cannot be meaningfully recommended even 1 day after the initial announcement. The time value of a recommendation is clearly dependent on the novelty and diversity of the recommended items. We still need more theoretical, methodological and algorithmic developments around these aspects. For instance, modeling feature-based novelty in probabilistic terms in order to unify discovery and familiarity models would be an interesting line for future work. Aspects such as the time dimension during which items may recover part of their novelty value, or the variability among users regarding their degree of novelty-seeking are examples of issues that require further research and are mentioned in Chap. 26.

Somewhat connected to the novelty and diversity topic is the issue of achieving an effective tradeoff between exploration and exploitation, which is touched upon in the active learning chapter (Chap. 24). This challenge refers to the fundamental dilemma that a designer must properly tackle, or decide whether to keep recommending items that the system can now identify as good recommendations, given the data currently available to the system, or to further explore user preferences (e.g., asking the user to rate additional, particular items) in order to build newer and possibly better recommendations in the future.

1.9.3 New Recommendation Tasks

The application of recommender systems is still dominated by solutions for recommending relatively simple and inexpensive products like movies, music, news and books. While there are systems managing more complex item types, such as financial investments or travel, these item categories are considered as atypical cases. Inevitably, complex domains require more elaborated solutions such as those based on knowledge and largely discussed in Chap. 5. Complex products are typically configurable or offered in several variants. This feature still poses a challenge to recommender systems, which are instead designed to consider different configurations as different items. Identifying the more suitable configuration

requires reasoning between the interactions of alternative configurations (classifying and grouping items) and calls for addressing the specificity of the human decision making task generated by the selection of a configuration. In general terms, addressing new types of recommendation domains can call for the introduction of many new and interesting research lines. For instance, Chap. 16 clearly shows how different the recommendation technique must be in domains where reciprocal recommendations are needed, as in dating applications.

As we already indicated in the first edition of this handbook, recommenders that optimize a sequence of recommendations, e.g., a new book every week, are not frequent and we believe that this is still an open issue. It is important to study the sequential dimension of users' decision making both within a recommendation session and between recommendation sessions. Here, we want to further note the importance of such a topic in group recommenders (see Chap. 22). In these systems, sequential recommendations are a natural setting, since stable groups, such as friends or families, repeatedly choose items of the same type, e.g., when deciding where to go for vacation or what to eat at home. A lot more research is needed on algorithms and user interfaces for producing coherent sequences of recommendations. In particular, one should model the effect on users of several contextual conditions such as the manner in which already-shown-items could influence the user evaluation of the next recommendations, or the social role and relationships of the group members.

As it is discussed in Chap. 14 most of the popular recommender systems are now accessed through mobile systems that follow their owners throughout their daily life, and are always within an arm's reach of their owners. In this scenario, as for instance in Google Now, recommender systems can proactively send notifications to their users about items of potential interest that are relevant because of the contextual situation of the user. The challenge is finding true relevant items for the user situation and not overburdening them with a stream of irrelevant interruptions. To address this goal, we must better exploit implicit feedback derived from user usage of the recommendations, but also learn to better identify contextual situations that require push recommendations. We believe that this depends on the detection of contextual changes that are significant to the user and therefore justify a recommendation. For instance, when it is the ideal time for a pause in writing a paper, i.e., the context is changing from work to leisure, a recommendation of a relevant, or personalized, article of sports news can be delivered. Understanding when context changes or could be forced to change and when a user may be receptive to a recommendation push is a challenging issue for further research. As it is also suggested in Chap. 6, in order to develop these new and compelling context aware systems, we need to explore novel engineering solutions to CARS, including: novel data structures, storage systems, user interface components and service oriented architectures.

Another task that we believe should be explored further is guided navigation. This refers to combining classical recommendation lists with tools that let the user navigate more autonomously in the space of possible options. User action interpretation refers to the possibility that in addition to explicit ratings, there could

be many more actions performed by the user operating the recommender that can be detected, analyzed and used to build a better prediction model. The idea is that every single user action should be exploited in the recommendation process. But it is challenging to interpret the user's actions, i.e., the intent behind an action, and there are actions that should be discarded because they were not produced by genuine users, such as actions performed by different users on the same browser, or false and malicious registrations or data or log data caused by robots or crawlers.

Finally, we want to note again that in order to deliver a small set of relevant recommendations to the user, correctly predicting the user rating is only one option. An alternative consists of predicting how the user would compare or rank the available options. This is nowadays an important line of research in recommender systems. Chapter 11 discusses this issue in the context of the Netflix recommender.

Finally, we hope that this handbook, as a useful tool for practitioners and researchers, will contribute to further developing knowledge in this exciting and useful research area and provide a baseline for further exploring the above mentioned issues. Currently the research on RSs has greatly benefited from the combined interest and efforts that industry and academia have invested in this field. We therefore wish the best to both groups as they read this handbook and we hope that it will attract even more researchers to work in this highly interesting and challenging field.

References

1. Adomavicius, G., Tuzhilin, A.: Personalization technologies: a process-oriented perspective. Commun. ACM **48**(10), 83–90 (2005)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17**(6), 734–749 (2005)
3. Amatriain, X.: Mining large streams of user data for personalized recommendations. SIGKDD Explor. Newsl. **14**(2), 37–48 (2013)
4. Arazy, O., Kumar, N., Shapira, B.: Improving social recommender systems. IT Professional **11**(4), 38–44 (2009)
5. Asoh, H., Ono, C., Habu, Y., Takasaki, H., Takenaka, T., Motomura, Y.: An acceptance model of recommender systems based on a large-scale internet survey. In: Advances in User Modeling - UMAP 2011 Workshops, Girona, Spain, July 11–15, 2011, Revised Selected Papers, pp. 410–414 (2011)
6. Bailey, R.A.: Design of comparative experiments. Cambridge University Press Cambridge (2008)
7. Balabanovic, M., Shoham, Y.: Content-based, collaborative recommendation. Communication of ACM **40**(3), 66–72 (1997)
8. Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Model. User-Adapt. Interact. **24**(1–2), 7–34 (2014)
9. Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisels, A., Shani, G., Naamani, L.: Recommender system from personal social networks. In: K. Wegrzyn-Wolska, P.S. Szczepaniak (eds.) AWIC, *Advances in Soft Computing*, vol. 43, pp. 47–55. Springer (2007)
10. Berkovsky, S., Kuflik, T., Ricci, F.: Mediation of user models for enhanced personalization in recommender systems. User Modeling and User-Adapted Interaction **18**(3), 245–286 (2008)

11. Berkovsky, S., Kuflik, T., Ricci, F.: Cross-representation mediation of user models. User Modeling and User-Adapted Interaction **19**(1–2), 35–63 (2009)
12. Billsus, D., Pazzani, M.: Learning probabilistic user models. In: UM97 Workshop on Machine Learning for User Modeling (1997). URL http://www.dfki.de/~bauer/um-ws/
13. Bobadilla, J., Ortega, F., Hernando, A., Gutierrez, A.: Recommender systems survey. Knowledge-Based Systems **46**(0), 109–132 (2013)
14. Borràs, J., Moreno, A., Valls, A.: Intelligent tourism recommender systems: A survey. Expert Systems with Applications **41**(16), 7370–7389 (2014)
15. Bridge, D., Göker, M., McGinty, L., Smyth, B.: Case-based recommender systems. The Knowledge Engineering review **20**(3), 315–320 (2006)
16. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6**(2–3), 87–129 (1996)
17. Burke, R.: Hybrid web recommender systems. In: The Adaptive Web, pp. 377–408. Springer Berlin / Heidelberg (2007)
18. Chen, L., de Gemmis, M., Felfernig, A., Lops, P., Ricci, F., Semeraro, G.: Human decision making and recommender systems. TiiS **3**(3), 17 (2013)
19. Chen, L., Pu, P.: Critiquing-based recommenders: survey and emerging trends. User Model. User-Adapt. Interact. **22**(1–2), 125–150 (2012)
20. Cosley, D., Lam, S.K., Albert, I., Konstant, J.A., Riedl, J.: Is seeing believing? how recommender system interfaces affect users' opinions. In: In Proceedings of the CHI 2003 Conference on Human factors in Computing Systems, pp. 585–592. Fort Lauderdale, FL (2003)
21. Ekstrand, M.D., Harper, F.M., Willemsen, M.C., Konstan, J.A.: User perception of differences in recommender algorithms. In: Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014, pp. 161–168 (2014)
22. Fisher, G.: User modeling in human-computer interaction. User Modeling and User-Adapted Interaction **11**, 65–86 (2001)
23. Golbeck, J.: Generating predictive movie recommendations from trust in social networks. In: Trust Management, 4th International Conference, iTrust 2006, Pisa, Italy, May 16–19, 2006, Proceedings, pp. 93–104 (2006)
24. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992)
25. Herlocker, J., Konstan, J., Riedl, J.: Explaining collaborative filtering recommendations. In: In proceedings of ACM 2000 Conference on Computer Supported Cooperative Work, pp. 241–250 (2000)
26. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transaction on Information Systems **22**(1), 5–53 (2004)
27. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2010)
28. Kaminskas, M., Ricci, F.: Contextual music information retrieval and recommendation: State of the art and challenges. Computer Science Review **6**(2–3), 89–119 (2012)
29. Knijnenburg, B.P., Willemsen, M.C., Gantner, Z., Soncu, H., Newell, C.: Explaining the user experience of recommender systems. User Modeling and User-Adapted Interaction **22**(4–5), 441–504 (2012). DOI 10.1007/s11257-011-9118-4
30. Konstan, J.A., Riedl, J.: Recommender systems: from algorithms to user experience. User Modeling and User-Adapted Interaction **22**(1–2), 101–123 (2012)
31. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8), 30–37 (2009)
32. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing **7**(1), 76–80 (2003)
33. Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. In: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (eds.) Recommender Systems Handbook, pp. 73–105. Springer Verlag (2011)

34. Lu, L., Medo, M., Yeung, C.H., Zhang, Y.C., Zhang, Z.K., Zhou, T.: Recommender systems. Physics Reports **519**(1), 1–49 (2012)
35. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, pp. 73–82. ACM (2009)
36. Mahmood, T., Ricci, F., Venturini, A.: Improving recommendation effectiveness by adapting the dialogue strategy in online travel planning. International Journal of Information Technology and Tourism **11**(4), 285–302 (2009)
37. McFee, B., Bertin-Mahieux, T., Ellis, D.P., Lanckriet, G.R.: The million song dataset challenge. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion, pp. 909–916. ACM, New York, NY, USA (2012)
38. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, pp. 1097–1101. ACM Press, New York, NY, USA (2006)
39. Montaner, M., López, B., de la Rosa, J.L.: A taxonomy of recommender agents on the internet. Artificial Intelligence Review **19**(4), 285–330 (2003)
40. Park, D.H., Kim, H.K., Choi, I.Y., Kim, J.K.: A literature review and classification of recommender systems research. Expert Systems with Applications **39**(11), 10,059–10,072 (2012)
41. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings ACM Conference on Computer-Supported Cooperative Work, pp. 175–186 (1994)
42. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40**(3), 56–58 (1997)
43. Ricci, F.: Travel recommender systems. IEEE Intelligent Systems **17**(6), 55–57 (2002)
44. Ricci, F.: Recommender systems: Models and techniques. In: Encyclopedia of Social Network Analysis and Mining, pp. 1511–1522. Springer (2014)
45. Ricci, F., Cavada, D., Mirzadeh, N., Venturini, A.: Case-based travel recommendations. In: D.R. Fesenmaier, K. Woeber, H. Werthner (eds.) Destination Recommendation Systems: Behavioural Foundations and Applications, pp. 67–93. CABI (2006)
46. Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T. (eds.): Recommendation Systems in Software Engineering. Springer (2014)
47. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, pp. 291–324. Springer Berlin / Heidelberg (2007)
48. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data Mining and Knowledge Discovery **5**(1/2), 115–153 (2001)
49. Schwartz, B.: The Paradox of Choice. ECCO, New York (2004)
50. van Setten, M., McNee, S.M., Konstan, J.A.: Beyond personalization: the next stage of recommender systems research. In: R.S. Amant, J. Riedl, A. Jameson (eds.) IUI, p. 8. ACM (2005)
51. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Proceedings of the Conference on Human Factors in Computing Systems (CHI'95), pp. 210–217 (1995)
52. Sinha, R.R., Swearingen, K.: Comparing recommendations made by online systems and friends. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries (2001)
53. Swearingen, K., Sinha, R.: Beyond algorithms: An HCI perspective on recommender systems. In: J.L. Herlocker (ed.) Recommender Systems, papers from the 2001 ACM SIGIR Workshop. New Orleans, LA - USA (2001)
54. Taghipour, N., Kardan, A., Ghidary, S.S.: Usage-based web recommendations: a reinforcement learning approach. In: Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys 2007, Minneapolis, MN, USA, October 19–20, 2007, pp. 113–120 (2007)
55. Verbert, K., Drachsler, H., Manouselis, N., Wolpers, M., Vuorikari, R., Duval, E.: Dataset-driven research for improving recommender systems for learning. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge, LAK '11, pp. 44–53. ACM, New York, NY, USA (2011)

Part I
Recommendation Techniques

Chapter 2
A Comprehensive Survey of Neighborhood-Based Recommendation Methods

Xia Ning, Christian Desrosiers, and George Karypis

2.1 Introduction

The appearance and growth of online markets has had a considerable impact on the habits of consumers, providing them access to a greater variety of products and information on these goods. While this freedom of purchase has made online commerce into a multi-billion dollar industry, it also made it more difficult for consumers to select the products that best fit their needs. One of the main solutions proposed for this information overload problem are recommender systems, which provide automated and personalized suggestions of products to consumers.

The recommendation problem can be defined as estimating the response of a user for new items, based on historical information stored in the system, and suggesting to this user *novel* and *original* items for which the predicted response is *high*. User-item responses can be numerical values known as ratings (e.g., 1–5 stars), ordinal values (e.g., strongly agree, agree, neutral, disagree, strongly disagree) representing the possible levels of user appreciation, or binary values (e.g., like/dislike or interested/not interested). Moreover, user responses can be obtained explicitly, for

X. Ning
Computer Science Department, Purdue University, West Lafayette, IN, USA
e-mail: xning@iupui.edu

C. Desrosiers (✉)
Software Engineering and IT Department, École de Technologie Supérieure, Montreal, QC, Canada
e-mail: christian.desrosiers@etsmtl.ca

G. Karypis
Computer Science and Engineering Department, University of Minnesota, Minneapolis, MN, USA
e-mail: karypis@cs.umn.edu

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
DOI 10.1007/978-1-4899-7637-6_2

instance, through ratings/reviews entered by users in the system, or implicitly, from purchase history or access patterns [39, 70]. For the purpose of simplicity, from this point on, we will call rating any type of user-item response.

Item recommendation approaches can be divided in two broad categories: personalized and non-personalized. Among the personalized approaches are *content-based* and *collaborative filtering* methods, as well as *hybrid* techniques combining these two types of methods. The general principle of content-based (or cognitive) methods [4, 8, 42, 54] is to identify the common characteristics of items that have received a favorable rating from a user, and then recommend to this user new items that share these characteristics. Recommender systems based purely on content generally suffer from the problems of *limited content analysis* and *over-specialization* [63]. Limited content analysis occurs when the system has a limited amount of information on its users or the content of its items. For instance, privacy issues might refrain a user from providing personal information, or the precise content of items may be difficult or costly to obtain for some types of items, such as music or images. Another problem is that the content of an item is often insufficient to determine its quality. Over-specialization, on the other hand, is a side effect of the way in which content-based systems recommend new items, where the predicted rating of a user for an item is high if this item is similar to the ones liked by this user. For example, in a movie recommendation application, the system may recommend to a user a movie of the same genre or having the same actors as movies already seen by this user. Because of this, the system may fail to recommend items that are different but still interesting to the user.

Instead of depending on content information, collaborative (or social) filtering approaches use the rating information of other users and items in the system. The key idea is that the rating of a target user for a new item is likely to be similar to that of another user, if both users have rated other items in a similar way. Likewise, the target user is likely to rate two items in a similar fashion, if other users have given similar ratings to these two items. Collaborative approaches overcome some of the limitations of content-based ones. For instance, items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users. Furthermore, collaborative recommendations are based on the quality of items as evaluated by peers, instead of relying on content that may be a bad indicator of quality. Finally, unlike content-based systems, collaborative filtering ones can recommend items with very different content, as long as other users have already shown interest for these different items.

Collaborative filtering approaches can be grouped in the two general classes of *neighborhood* and *model*-based methods. In neighborhood-based (memory-based [10] or heuristic-based [2]) collaborative filtering [14, 15, 27, 39, 44, 48, 57, 59, 63], the user-item ratings stored in the system are directly used to predict ratings for new items. This can be done in two ways known as *user-based* or *item-based* recommendation. User-based systems, such as GroupLens [39], Bellcore video [27], and Ringo [63], evaluate the interest of a target user for an item using the ratings for this item by other users, called *neighbors*, that have similar rating patterns. The neighbors of the target user are typically the users whose ratings are most correlated

to the target user's ratings. Item-based approaches [15, 44, 59], on the other hand, predict the rating of a user for an item based on the ratings of the user for similar items. In such approaches, two items are similar if several users of the system have rated these items in a similar fashion.

In contrast to neighborhood-based systems, which use the stored ratings directly in the prediction, model-based approaches use these ratings to learn a predictive model. Salient characteristics of users and items are captured by a set of model parameters, which are learned from training data and later used to predict new ratings. Model-based approaches for the task of recommending items are numerous and include Bayesian Clustering [10], Latent Semantic Analysis [28], Latent Dirichlet Allocation [9], Maximum Entropy [72], Boltzmann Machines [58], Support Vector Machines [23], and Singular Value Decomposition [6, 40, 53, 68, 69]. A survey of state-of-the-art model-based methods can be found in Chap. 3 of this book.

Finally, to overcome certain limitations of content-based and collaborative filtering methods, hybrid recommendation approaches combine characteristics of both types of methods. Content-based and collaborative filtering methods can be combined in various ways, for instance, by merging their individual predictions into a single, more robust prediction [8, 55], or by adding content information into a collaborative filtering model [1, 3, 51, 65, 71]. Several studies have shown hybrid recommendation approaches to provide more accurate recommendations than pure content-based or collaborative methods, especially when few ratings are available [2].

2.1.1 *Advantages of Neighborhood Approaches*

While recent investigations show state-of-the-art model-based approaches superior to neighborhood ones in the task of predicting ratings [40, 67], there is also an emerging understanding that good prediction accuracy alone does not guarantee users an effective and satisfying experience [26]. Another factor that has been identified as playing an important role in the appreciation of users for the recommender system is *serendipity* [26, 59]. Serendipity extends the concept of novelty by helping a user find an interesting item he or she might not have otherwise discovered. For example, recommending to a user a movie directed by his favorite director constitutes a novel recommendation if the user was not aware of that movie, but is likely not serendipitous since the user would have discovered that movie on his own.

Model-based approaches excel at characterizing the preferences of a user with latent factors. For example, in a movie recommender system, such methods may determine that a given user is a fan of movies that are both funny and romantic, without having to actually define the notions "funny" and "romantic". This system would be able to recommend to the user a romantic comedy that may not have been known to this user. However, it may be difficult for this system to recommend

a movie that does not quite fit this high-level genre, for instance, a funny parody of horror movies. Neighborhood approaches, on the other hand, capture local associations in the data. Consequently, it is possible for a movie recommender system based on this type of approach to recommend the user a movie very different from his usual taste or a movie that is not well known (e.g. repertoire film), if one of his closest neighbors has given it a strong rating. This recommendation may not be a guaranteed success, as would be a romantic comedy, but it may help the user discover a whole new genre or a new favorite actor/director.

The main advantages of neighborhood-based methods are:

- **Simplicity:** Neighborhood-based methods are intuitive and relatively simple to implement. In their simplest form, only one parameter (the number of neighbors used in the prediction) requires tuning.
- **Justifiability:** Such methods also provide a concise and intuitive justification for the computed predictions. For example, in item-based recommendation, the list of neighbor items, as well as the ratings given by the user to these items, can be presented to the user as a justification for the recommendation. This can help the user better understand the recommendation and its relevance, and could serve as basis for an interactive system where users can select the neighbors for which a greater importance should be given in the recommendation [6].
- **Efficiency:** One of the strong points of neighborhood-based systems are their efficiency. Unlike most model-based systems, they require no costly training phases, which need to be carried at frequent intervals in large commercial applications. These systems may require pre-computing nearest neighbors in an offline step, which is typically much cheaper than model training, providing near instantaneous recommendations. Moreover, storing these nearest neighbors requires very little memory, making such approaches scalable to applications having millions of users and items.
- **Stability:** Another useful property of recommender systems based on this approach is that they are little affected by the constant addition of users, items and ratings, which are typically observed in large commercial applications. For instance, once item similarities have been computed, an item-based system can readily make recommendations to new users, without having to re-train the system. Moreover, once a few ratings have been entered for a new item, only the similarities between this item and the ones already in the system need to be computed.

While neighborhood-based methods have gained popularity due to these advantages, they are also known to suffer from the problem of limited coverage, which causes some items to be never recommended. Also, traditional methods of this category are known to be more sensitive to the sparseness of ratings and the cold-start problem, where the system has only a few ratings, or no rating at all, for new users and items. Section 2.5 presents more advanced neighborhood-based techniques that can overcome these problems.

2.1.2 Objectives and Outline

This chapter has two main objectives. It first serves as a general guide on neighborhood-based recommender systems, and presents practical information on how to implement such recommendation approaches. In particular, the main components of neighborhood-based methods will be described, as well as the benefits of the most common choices for each of these components. Secondly, it presents more specialized techniques on the subject that address particular aspects of recommending items, such as data sparsity. Although such techniques are not required to implement a simple neighborhood-based system, having a broader view of the various difficulties and solutions for neighborhood methods may help making appropriate decisions during the implementation process.

The rest of this document is structured as follows. In Sect. 2.2, we first give a formal definition of the item recommendation task and present the notation used throughout the chapter. In Sect. 2.3, the principal neighborhood approaches, predicting user ratings for new items based on regression or classification, are then introduced, and the main advantages and flaws of these approaches are described. This section also presents two complementary ways of implementing such approaches, either based on user or item similarities, and analyzes the impact of these two implementations on the accuracy, efficiency, stability, justifiability and serendipity of the recommender system. Section 2.4, on the other hand, focuses on the three main components of neighborhood-based recommendation methods: rating normalization, similarity weight computation, and neighborhood selection. For each of these components, the most common approaches are described, and their respective benefits compared. In Sect. 2.5, the problems of limited coverage and data sparsity are introduced, and several solutions proposed to overcome these problems are described. In particular, several techniques based on dimensionality reduction and graphs are presented. Finally, the last section of this document summarizes the principal characteristics and methods of neighborhood-based recommendation, and gives a few more pointers on implementing such methods.

2.2 Problem Definition and Notation

In order to give a formal definition of the item recommendation task, we introduce the following notation. The set of users in the recommender system will be denoted by $\mathcal{U}$, and the set of items by $\mathcal{I}$. Moreover, we denote by $\mathcal{R}$ the set of ratings recorded in the system, and write $\mathcal{S}$ the set of possible values for a rating (e.g., $\mathcal{S} = [1, 5]$ or $\mathcal{S} = \{\text{like}, \text{dislike}\}$). Also, we suppose that no more than one rating can be made by any user $u \in \mathcal{U}$ for a particular item $i \in \mathcal{I}$ and write r_{ui} this rating. To identify the subset of users that have rated an item i, we use the notation $\mathcal{U}_i$. Likewise, $\mathcal{I}_u$ represents the subset of items that have been rated by a user u. Finally, the items that

have been rated by two users u and v, i.e. $\mathcal{I}_u \cap \mathcal{I}_v$, is an important concept in our presentation, and we use $\mathcal{I}_{uv}$ to denote this concept. In a similar fashion, $\mathcal{U}_{ij}$ is used to denote the set of users that have rated both items i and j.

Two of the most important problems associated with recommender systems are the *rating prediction* and *top-N* recommendation problems. The first problem is to predict the rating that a user u will give his or her unrated item i. When ratings are available, this task is most often defined as a regression or (multi-class) classification problem where the goal is to learn a function $f : \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{S}$ that predicts the rating $f(u, i)$ of a user u for a new item i. Accuracy is commonly used to evaluate the performance of the recommendation method. Typically, the ratings $\mathcal{R}$ are divided into a *training* set $\mathcal{R}_{\text{train}}$ used to learn f, and a *test* set $\mathcal{R}_{\text{test}}$ used to evaluate the prediction accuracy. Two popular measures of accuracy are the *Mean Absolute Error* (MAE):

$$\text{MAE}(f) = \frac{1}{|\mathcal{R}_{\text{test}}|} \sum_{r_{ui} \in \mathcal{R}_{\text{test}}} |f(u, i) - r_{ui}|, \tag{2.1}$$

and the *Root Mean Squared Error* (RMSE):

$$\text{RMSE}(f) = \sqrt{\frac{1}{|\mathcal{R}_{\text{test}}|} \sum_{r_{ui} \in \mathcal{R}_{\text{test}}} (f(u, i) - r_{ui})^2}. \tag{2.2}$$

When ratings are not available, for instance, if only the list of items purchased by each user is known, measuring the rating prediction accuracy is not possible. In such cases, the problem of finding the best item is usually transformed into the task of recommending to an active user u_a a list $L(u_a)$ containing N items likely to interest him or her [15, 59]. The quality of such method can be evaluated by splitting the items of $\mathcal{I}$ into a set $\mathcal{I}_{\text{train}}$, used to learn L, and a test set $\mathcal{I}_{\text{test}}$. Let $T(u) \subset \mathcal{I}_u \cap \mathcal{I}_{\text{test}}$ be the subset of test items that a user u found relevant. If the user responses are binary, these can be the items that u has rated positively. Otherwise, if only a list of purchased or accessed items is given for each user u, then these items can be used as $T(u)$. The performance of the method is then computed using the measures of *precision* and *recall*:

$$\text{Precision}(L) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} |L(u) \cap T(u)| \,/\, |L(u)| \tag{2.3}$$

$$\text{Recall}(L) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} |L(u) \cap T(u)| \,/\, |T(u)|. \tag{2.4}$$

A drawback of this task is that all items of a recommendation list $L(u)$ are considered equally interesting to user u. An alternative setting, described in [15], consists in learning a function L that maps each user u to a list $L(u)$ where items are *ordered*

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv}\, h(r_{vi})}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}\right). \tag{2.8}$$

Note that the predicted rating must be converted back to the original scale, hence the h^{-1} in the equation. The most common approaches to normalize ratings will be presented in Sect. 2.4.1.

2.3.2 User-Based Classification

The prediction approach just described, where the predicted ratings are computed as a weighted average of the neighbors' ratings, essentially solves a *regression* problem. Neighborhood-based *classification*, on the other hand, finds the most likely rating given by a user u to an item i, by having the nearest-neighbors of u vote on this value. The vote v_{ir} given by the k-NN of u for the rating $r \in \mathcal{S}$ can be obtained as the sum of the similarity weights of neighbors that have given this rating to i:

$$v_{ir} = \sum_{v \in \mathcal{N}_i(u)} \delta(r_{vi} = r)\, w_{uv}, \tag{2.9}$$

where $\delta(r_{vi} = r)$ is 1 if $r_{vi} = r$, and 0 otherwise. Once this has been computed for every possible rating value, the predicted rating is simply the value r for which v_{ir} is the greatest.

Example 2.3. Suppose once again that the two nearest-neighbors of Eric are Lucy and Diane with respective similarity weights 0.75 and 0.15. In this case, ratings 5 and 3 each have one vote. However, since Lucy's vote has a greater weight than Diane's, the predicted rating will be $\hat{r} = 5$.

A classification method that considers normalized ratings can also be defined. Let $\mathcal{S}'$ be the set of possible normalized values (that may require discretization), the predicted rating is obtained as:

$$\hat{r}_{ui} = h^{-1}\left(\arg\max_{r \in \mathcal{S}'} \sum_{v \in \mathcal{N}_i(u)} \delta(h(r_{vi}) = r)\, w_{uv}\right). \tag{2.10}$$

2.3.3 Regression vs Classification

The choice between implementing a neighborhood-based regression or classification method largely depends on the system's rating scale. Thus, if the rating scale

is continuous, e.g. ratings in the *Jester* joke recommender system [20] can take any value between −10 and 10, then a regression method is more appropriate. On the contrary, if the rating scale has only a few discrete values, e.g. "good" or "bad", or if the values cannot be ordered in an obvious fashion, then a classification method might be preferable. Furthermore, since normalization tends to map ratings to a continuous scale, it may be harder to handle in a classification approach.

Another way to compare these two approaches is by considering the situation where all neighbors have the same similarity weight. As the number of neighbors used in the prediction increases, the rating r_{ui} predicted by the regression approach will tend toward the mean rating of item i. Suppose item i has only ratings at either end of the rating range, i.e. it is either loved or hated, then the regression approach will make the safe decision that the item's worth is average. This is also justified from a statistical point of view since the expected rating (estimated in this case) is the one that minimizes the RMSE. On the other hand, the classification approach will predict the rating as the most frequent one given to i. This is more risky as the item will be labeled as either "good" or "bad". However, as mentioned before, taking risks may be desirable if it leads to serendipitous recommendations.

2.3.4 Item-Based Recommendation

While user-based methods rely on the opinion of like-minded users to predict a rating, item-based approaches [15, 44, 59] look at ratings given to similar items. Let us illustrate this approach with our toy example.

Example 2.4. Instead of consulting with his peers, Eric instead determines whether the movie "Titanic" is right for him by considering the movies that he has already seen. He notices that people that have rated this movie have given similar ratings to the movies "Forrest Gump" and "Wall-E". Since Eric liked these two movies he concludes that he will also like the movie "Titanic".

This idea can be formalized as follows. Denote by $\mathcal{N}_u(i)$ the items rated by user u most similar to item i. The predicted rating of u for i is obtained as a weighted average of the ratings given by u to the items of $\mathcal{N}_u(i)$:

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij}\, r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|}. \tag{2.11}$$

Example 2.5. Suppose our prediction is again made using two nearest-neighbors, and that the items most similar to "Titanic" are "Forrest Gump" and "Wall-E", with respective similarity weights 0.85 and 0.75. Since ratings of 5 and 4 were given by Eric to these two movies, the predicted rating is computed as

$$\hat{r} = \frac{0.85\times5 + 0.75\times4}{0.85 + 0.75} \simeq 4.53.$$

Again, the differences in the users' individual rating scales can be considered by normalizing ratings with a function h:

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{j\in\mathcal{N}_u(i)} w_{ij}\, h(r_{uj})}{\sum_{j\in\mathcal{N}_u(i)} |w_{ij}|}\right). \tag{2.12}$$

Moreover, we can also define an item-based classification approach. In this case, the items j rated by user u vote for the rating to be given to a new item i, and these votes are weighted by the similarity between i and j. The normalized version of this approach can be expressed as follows:

$$\hat{r}_{ui} = h^{-1}\left(\arg\max_{r\in\mathcal{S}'} \sum_{j\in\mathcal{N}_u(i)} \delta(h(r_{uj}) = r)\, w_{ij}\right). \tag{2.13}$$

2.3.5 User-Based vs Item-Based Recommendation

When choosing between the implementation of a user-based and an item-based neighborhood recommender system, five criteria should be considered:

- **Accuracy:** The accuracy of neighborhood recommendation methods depends mostly on the ratio between the number of users and items in the system. As will be presented in Sect. 2.4.2, the similarity between two users in user-based methods, which determines the neighbors of a user, is normally obtained by comparing the ratings made by these users on the same items. Consider a system that has 10,000 ratings made by 1000 users on 100 items, and suppose, for the purpose of this analysis, that the ratings are distributed uniformly over the items.[1] Following Table 2.1, the average number of users available as potential neighbors is roughly 650. However, the average number of common ratings used to compute the similarities is only 1. On the other hand, an item-based method usually computes the similarity between two items by comparing ratings made by the same user on these items. Assuming once more a uniform distribution of ratings, we find an average number of potential neighbors of 99 and an average number of ratings used to compute the similarities of 10.

[1] The distribution of ratings in real-life data is normally skewed, i.e. most ratings are given to a small proportion of items.

Table 2.1 The average number of neighbors and average number of ratings used in the computation of similarities for user-based and item-based neighborhood methods

	Average neighbors	Average ratings
User-based	$(\lvert\mathcal{U}\rvert - 1)\left(1 - \left(\frac{\lvert\mathcal{I}\rvert - p}{\lvert\mathcal{I}\rvert}\right)^p\right)$	$\frac{p^2}{\lvert\mathcal{I}\rvert}$
Item-based	$(\lvert\mathcal{I}\rvert - 1)\left(1 - \left(\frac{\lvert\mathcal{U}\rvert - q}{\lvert\mathcal{U}\rvert}\right)^q\right)$	$\frac{q^2}{\lvert\mathcal{U}\rvert}$

A uniform distribution of ratings is assumed with average number of ratings per user $p = |\mathcal{R}|/|\mathcal{U}|$, and average number of ratings per item $q = |\mathcal{R}|/|\mathcal{I}|$

Table 2.2 The space and time complexity of user-based and item-based neighborhood methods, as a function of the maximum number of ratings per user $p = \max_u |\mathcal{I}_u|$, the maximum number of ratings per item $q = \max_i |\mathcal{U}_i|$, and the maximum number of neighbors used in the rating predictions k

	Space	Time	
		Training	Online
User-based	$O(\lvert\mathcal{U}\rvert^2)$	$O(\lvert\mathcal{U}\rvert^2 p)$	$O(\lvert\mathcal{I}\rvert k)$
Item-based	$O(\lvert\mathcal{I}\rvert^2)$	$O(\lvert\mathcal{I}\rvert^2 q)$	$O(\lvert\mathcal{I}\rvert k)$

In general, a small number of high-confidence neighbors is by far preferable to a large number of neighbors for which the similarity weights are not trustable. In cases where the number of users is much greater than the number of items, such as large commercial systems like *Amazon.com*, item-based methods can therefore produce more accurate recommendations [16, 59]. Likewise, systems that have less users than items, e.g., a research paper recommender with thousands of users but hundreds of thousands of articles to recommend, may benefit more from user-based neighborhood methods [26].

- **Efficiency:** As shown in Table 2.2, the memory and computational efficiency of recommender systems also depends on the ratio between the number of users and items. Thus, when the number of users exceeds the number of items, as is it most often the case, item-based recommendation approaches require much less memory and time to compute the similarity weights (training phase) than user-based ones, making them more scalable. However, the time complexity of the online recommendation phase, which depends only on the number of available items and the maximum number of neighbors, is the same for user-based and item-based methods.

  In practice, computing the similarity weights is much less expensive than the worst-case complexity reported in Table 2.2, due to the fact that users rate only a few of the available items. Accordingly, only the non-zero similarity weights need to be stored, which is often much less than the number of user pairs. This number can be further reduced by storing for each user only the top N weights, where N is a parameter [59] that is sufficient for satisfactory coverage on user-item pairs. In the same manner, the non-zero weights can be computed efficiently without having to test each pair of users or items, which makes neighborhood methods scalable to very large systems.

- **Stability:** The choice between a user-based and an item-based approach also depends on the frequency and amount of change in the users and items of the system. If the list of available items is fairly static in comparison to the users of the system, an item-based method may be preferable since the item similarity weights could then be computed at infrequent time intervals while still being able to recommend items to new users. On the contrary, in applications where the list of available items is constantly changing, e.g., an online article recommender, user-based methods could prove to be more stable.
- **Justifiability:** An advantage of item-based methods is that they can easily be used to justify a recommendation. Hence, the list of neighbor items used in the prediction, as well as their similarity weights, can be presented to the user as an explanation of the recommendation. By modifying the list of neighbors and/or their weights, it then becomes possible for the user to participate interactively in the recommendation process. User-based methods, however, are less amenable to this process because the active user does not know the other users serving as neighbors in the recommendation.
- **Serendipity:** In item-based methods, the rating predicted for an item is based on the ratings given to similar items. Consequently, recommender systems using this approach will tend to recommend to a user items that are related to those usually appreciated by this user. For instance, in a movie recommendation application, movies having the same genre, actors or director as those highly rated by the user are likely to be recommended. While this may lead to safe recommendations, it does less to help the user discover different types of items that he might like as much.

  Because they work with user similarity, on the other hand, user-based approaches are more likely to make serendipitous recommendations. This is particularly true if the recommendation is made with a small number of nearest-neighbors. For example, a user A that has watched only comedies may be very similar to a user B only by the ratings made on such movies. However, if B is fond of a movie in a different genre, this movie may be recommended to A through his similarity with B.

2.4 Components of Neighborhood Methods

In the previous section, we have seen that deciding between a regression and a classification rating prediction method, as well as choosing between a user-based or item-based recommendation approach, can have a significant impact on the accuracy, efficiency and overall quality of the recommender system. In addition to these crucial attributes, three very important considerations in the implementation of a neighborhood-based recommender system are (1) the normalization of ratings, (2) the computation of the similarity weights, and (3) the selection of neighbors. This section reviews some of the most common approaches for these three components, describes the main advantages and disadvantages of using each one of them, and gives indications on how to implement them.

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_u}{\sigma_u}.$$

A user-based prediction of rating r_{ui} using this normalization approach would therefore be obtained as

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \frac{\sum\limits_{v \in \mathcal{N}_i(u)} w_{uv} (r_{vi} - \bar{r}_v)/\sigma_v}{\sum\limits_{v \in \mathcal{N}_i(u)} |w_{uv}|}. \tag{2.16}$$

Likewise, the z-score normalization of r_{ui} in item-based methods divides the *item*-mean-centered rating by the standard deviation of ratings given to item i:

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_i}{\sigma_i}.$$

The item-based prediction of rating r_{ui} would then be

$$\hat{r}_{ui} = \bar{r}_i + \sigma_i \frac{\sum\limits_{j \in \mathcal{N}_u(i)} w_{ij} (r_{uj} - \bar{r}_j)/\sigma_j}{\sum\limits_{j \in \mathcal{N}_u(i)} |w_{ij}|}. \tag{2.17}$$

2.4.1.3 Choosing a Normalization Scheme

In some cases, rating normalization can have undesirable effects. For instance, imagine the case of a user that gave only the highest ratings to the items he has purchased. Mean-centering would consider this user as "easy to please" and any rating below this highest rating (whether it is a positive or negative rating) would be considered as negative. However, it is possible that this user is in fact "hard to please" and carefully selects only items that he will like for sure. Furthermore, normalizing on a few ratings can produce unexpected results. For example, if a user has entered a single rating or a few identical ratings, his rating standard deviation will be 0, leading to undefined prediction values. Nevertheless, if the rating data is not overly sparse, normalizing ratings has been found to consistently improve the predictions [25, 29].

Comparing mean-centering with Z-score, as mentioned, the second one has the additional benefit of considering the variance in the ratings of individual users or items. This is particularly useful if the rating scale has a wide range of discrete values or if it is continuous. On the other hand, because the ratings are divided and multiplied by possibly very different standard deviation values, Z-score can be more sensitive than mean-centering and, more often, predict ratings that are outside the rating scale. Lastly, while an initial investigation found mean-centering and Z-score to give comparable results [25], a more recent one showed Z-score to have more significant benefits [29].

Finally, if rating normalization is not possible or does not improve the results, another possible approach to remove the problems caused by the rating scale variance is *preference-based filtering*. The particularity of this approach is that it focuses on predicting the relative preferences of users instead of absolute rating values. Since an item preferred to another one remains so regardless of the rating scale, predicting relative preferences removes the need to normalize the ratings. More information on this approach can be found in [12, 18, 32, 33].

2.4.2 Similarity Weight Computation

The similarity weights play a double role in neighborhood-based recommendation methods: (1) they allow to select trusted neighbors whose ratings are used in the prediction, and (2) they provide the means to give more or less importance to these neighbors in the prediction. The computation of the similarity weights is one of the most critical aspects of building a neighborhood-based recommender system, as it can have a significant impact on both its accuracy and its performance.

2.4.2.1 Correlation-Based Similarity

A measure of the similarity between two objects a and b, often used in information retrieval, consists in representing these objects in the form of a vector $\mathbf{x}_a$ and $\mathbf{x}_b$ and computing the *Cosine Vector* (CV) (or *Vector Space*) similarity [4, 8, 42] between these vectors:

$$\cos(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^\top \mathbf{x}_b}{||\mathbf{x}_a|| ||\mathbf{x}_b||}.$$

In the context of item recommendation, this measure can be employed to compute user similarities by considering a user u as a vector $\mathbf{x}_u \in \mathbb{R}^{|I|}$, where $\mathbf{x}_{ui} = r_{ui}$ if user u has rated item i, and 0 otherwise. The similarity between two users u and v would then be computed as

$$CV(u, v) = \cos(\mathbf{x}_u, \mathbf{x}_v) = \frac{\sum_{i \in \mathcal{I}_{uv}} r_{ui}\, r_{vi}}{\sqrt{\sum_{i \in \mathcal{I}_u} r_{ui}^2 \sum_{j \in \mathcal{I}_v} r_{vj}^2}}, \tag{2.18}$$

where I_{uv} once more denotes the items rated by both u and v. A problem with this measure is that is does not consider the differences in the mean and variance of the ratings made by users u and v.

A popular measure that compares ratings where the effects of mean and variance have been removed is the *Pearson Correlation* (PC) similarity:

$$\mathrm{PC}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{vi} - \bar{r}_v)^2}}. \tag{2.19}$$

Note that this is different from computing the CV similarity on the Z-score normalized ratings, since the standard deviation of the ratings in evaluated only on the common items I_{uv}, not on the entire set of items rated by u and v, i.e. $\mathcal{I}_u$ and $\mathcal{I}_v$. The same idea can be used to obtain similarities between two items i and j [15, 59], this time by comparing the ratings made by users that have rated both these items:

$$\mathrm{PC}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_j)^2}}. \tag{2.20}$$

While the sign of a similarity weight indicates whether the correlation is direct or inverse, its magnitude (ranging from 0 to 1) represents the strength of the correlation.

Example 2.7. The similarities between the pairs of users and items of our toy example, as computed using PC similarity, are shown in Fig. 2.3. We can see that Lucy's taste in movies is very close to Eric's (similarity of 0.922) but very different from John's (similarity of −0.938). This means that Eric's ratings can be trusted to predict Lucy's, and that Lucy should discard John's opinion on movies or consider the opposite. We also find that the people that like "The Matrix" also like "Die Hard" but hate "Wall-E". Note that these relations were discovered without having any knowledge of the genre, director or actors of these movies.

The differences in the rating scales of individual users are often more pronounced than the differences in ratings given to individual items. Therefore, while computing the item similarities, it may be more appropriate to compare ratings that are centered on their *user* mean, instead of their *item* mean. The *Adjusted Cosine* (AC) similarity [59], is a modification of the PC item similarity which compares user-mean-centered ratings:

$$AC(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_u)^2}}.$$

In some cases, AC similarity has been found to outperform PC similarity on the prediction of ratings using an item-based method [59].

User-based Pearson correlation

	John	Lucy	Eric	Diane
John	1.000	-0.938	-0.839	0.659
Lucy	-0.938	1.000	0.922	-0.787
Eric	-0.839	0.922	1.000	-0.659
Diane	0.659	-0.787	-0.659	1.000

Item-based Pearson correlation

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
Matrix	1.000	-0.943	0.882	-0.974	-0.977
Titanic	-0.943	1.000	-0.625	0.931	0.994
Die Hard	0.882	-0.625	1.000	-0.804	-1.000
ForrestGump	-0.974	0.931	-0.804	1.000	0.930
Wall-E	-0.977	0.994	-1.000	0.930	1.000

Fig. 2.3 The *user* and *item* PC similarity for the ratings of Fig. 2.1

2.4.2.2 Other Similarity Measures

Several other measures have been proposed to compute similarities between users or items. One of them is the *Mean Squared Difference* (MSD) [63], which evaluate the similarity between two users u and v as the inverse of the average squared difference between the ratings given by u and v on the same items:

$$\text{MSD}(u, v) = \frac{|\mathcal{I}_{uv}|}{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - r_{vi})^2}. \tag{2.21}$$

While it could be modified to compute the differences on normalized ratings, the MSD similarity is limited compared to PC similarity because it does not allow to capture negative correlations between user preferences or the appreciation of different items. Having such negative correlations may improve the rating prediction accuracy [24].

Another well-known similarity measure is the *Spearman Rank Correlation* (SRC) [36]. While PC uses the rating values directly, SRC instead considers the ranking of these ratings. Denote by k_{ui} the rating rank of item i in user u's list of rated items (tied ratings get the average rank of their spot). The SRC similarity between two users u and v is evaluated as:

Table 2.3 The rating prediction accuracy (MAE) obtained on the *MovieLens* dataset using the mean squared difference (MSD), Spearman rank correlation and Pearson correlation (PC) similarity measures

k	MSD	SRC	PC
5	0.7898	0.7855	0.7829
10	0.7718	0.7636	0.7618
20	0.7634	0.7558	0.7545
60	0.7602	0.7529	0.7518
80	0.7605	0.7531	0.7523
100	0.7610	0.7533	0.7528

Results are shown for predictions using an increasing number of neighbors k

$$\text{SRC}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (k_{ui} - \overline{k}_u)(k_{vi} - \overline{k}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (k_{ui} - \overline{k}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (k_{vi} - \overline{k}_v)^2}}, \tag{2.22}$$

where $\overline{k}_u$ is the average rank of items rated by u.

The principal advantage of SRC is that it avoids the problem of rating normalization, described in the last section, by using rankings. On the other hand, this measure may not be the best one when the rating range has only a few possible values, since that would create a large number of tied ratings. Moreover, this measure is typically more expensive than PC as ratings need to be sorted in order to compute their rank.

Table 2.3 shows the user-based prediction accuracy (MAE) obtained with MSD, SRC and PC similarity measures, on the *MovieLens*[2] dataset [24]. Results are given for different values of k, which represents the maximum number of neighbors used in the predictions. For this data, we notice that MSD leads to the least accurate predictions, possibly due to the fact that it does not take into account negative correlations. Also, these results show PC to be slightly more accurate than SRC. Finally, although PC has been generally recognized as the best similarity measure, see e.g. [24], a more recent investigation has shown that the performance of such measure depended greatly on the data [29].

2.4.2.3 Considering the Significance of Weights

Because the rating data is frequently sparse in comparison to the number of users and items of a system, it is often the case that similarity weights are computed using only a few ratings given to common items or made by the same users. For example, if the system has 10,000 ratings made by 1000 users on 100 items (assuming a uniform distribution of ratings), Table 2.1 shows us that the similarity between two users is computed, on average, by comparing the ratings given by these users to a

[2] http://www.grouplens.org/.

similarity metric and their impact on the prediction accuracy are described in [5]. Note, however, that this model may require to recompute the similarity weights for each predicted rating, making it less suitable for online recommender systems.

2.4.3 Neighborhood Selection

The number of nearest-neighbors to select and the criteria used for this selection can also have a serious impact on the quality of the recommender system. The selection of the neighbors used in the recommendation of items is normally done in two steps: (1) a global filtering step where only the most likely candidates are kept, and (2) a per prediction step which chooses the best candidates for this prediction.

2.4.3.1 Pre-filtering of Neighbors

In large recommender systems that can have millions of users and items, it is usually not possible to store the (non-zero) similarities between each pair of users or items, due to memory limitations. Moreover, doing so would be extremely wasteful as only the most significant of these values are used in the predictions. The pre-filtering of neighbors is an essential step that makes neighborhood-based approaches practicable by reducing the amount of similarity weights to store, and limiting the number of candidate neighbors to consider in the predictions. There are several ways in which this can be accomplished:

- **Top-N filtering:** For each user or item, only a list of the N nearest-neighbors and their respective similarity weight is kept. To avoid problems with efficiency or accuracy, N should be chosen carefully. Thus, if N is too large, an excessive amount of memory will be required to store the neighborhood lists and predicting ratings will be slow. On the other hand, selecting a too small value for N may reduce the coverage of the recommendation method, which causes some items to be never recommended.
- **Threshold filtering:** Instead of keeping a fixed number of nearest-neighbors, this approach keeps all the neighbors whose similarity weight's magnitude is greater than a given threshold $w_{\min}$. While this is more flexible than the previous filtering technique, as only the most significant neighbors are kept, the right value of $w_{\min}$ may be difficult to determine.
- **Negative filtering:** In general, negative rating correlations are less reliable than positive ones. Intuitively, this is because strong positive correlation between two users is a good indicator of their belonging to a common group (e.g., teenagers, science-fiction fans, etc.). However, although negative correlation may indicate membership to different groups, it does not tell how different are these groups, or whether these groups are compatible for some other categories of items.

While experimental investigations [25, 26] have found negative correlations to provide no significant improvement in the prediction accuracy, whether such correlations can be discarded depends on the data.

Note that these three filtering approaches are not exclusive and can be combined to fit the needs of the recommender system. For instance, one could discard all negative similarities *as well as* those that are not in the top-N lists.

2.4.3.2 Neighbors in the Predictions

Once a list of candidate neighbors has been computed for each user or item, the prediction of new ratings is normally made with the k-nearest-neighbors, that is, the k neighbors whose similarity weight has the greatest magnitude. The choice of k can also have a significant impact on the accuracy and performance of the system.

As shown in Table 2.3, the prediction accuracy observed for increasing values of k typically follows a *concave* function. Thus, when the number of neighbors is restricted by using a small k (e.g., $k < 20$), the prediction accuracy is normally low. As k increases, more neighbors contribute to the prediction and the variance introduced by individual neighbors is averaged out. As a result, the prediction accuracy improves. Finally, the accuracy usually drops when too many neighbors are used in the prediction (e.g., $k > 50$), due to the fact that the few strong local relations are "diluted" by the many weak ones. Although a number of neighbors between 20 to 50 is most often described in the literature, see e.g. [24, 26], the optimal value of k should be determined by cross-validation.

On a final note, more serendipitous recommendations may be obtained at the cost of a decrease in accuracy, by basing these recommendations on a few very similar users. For example, the system could find the user most similar to the active one and recommend the new item that has received the highest rated from this user.

2.5 Advanced Techniques

The neighborhood approaches based on rating correlation, such as the ones presented in the previous sections, have two important flaws:

- **Limited coverage:** Because rating correlation measures the similarity between two users by comparing their ratings for the same items, users can be neighbors *only if* they have rated common items. This assumption is very limiting, as users having rated a few or no common items may still have similar preferences. Moreover, since only items rated by neighbors can be recommended, the coverage of such methods can also be limited. This limitation also applies to item-based systems, when two items have only a few or no co-ratings.
- **Sensitivity to sparse data:** Another consequence of rating correlation, addressed briefly in Sect. 2.3.5, is the fact that the accuracy of neighborhood-based

recommendation methods suffers from the lack of available ratings. Sparsity is a problem common to most recommender systems due to the fact that users typically rate only a small proportion of the available items [7, 21, 60, 61]. This is aggravated by the fact that users or items newly added to the system may have no ratings at all, a problem known as *cold-start* [62]. When the rating data is sparse, two users or items are unlikely to have common ratings and, consequently, neighborhood-based approaches will predict ratings using a very limited number of neighbors. Moreover, similarity weights may be computed using only a small number of ratings, resulting in biased recommendations (see Sect. 2.4.2.3 for this problem).

A common solution for these problems is to fill the missing ratings with default values [10, 15], such as the middle value of the rating range, or the average user or item rating. A more reliable approach is to use content information to fill out the missing ratings [13, 21, 39, 47]. For instance, the missing ratings can be provided by autonomous agents called *filterbots* [21, 39], that act as ordinary users of the system and rate items based on some specific characteristics of their content. The missing ratings can instead be predicted by a content-based approach [47]. Furthermore, content similarity can also be used "instead of" or "in addition to" rating correlation similarity to find the nearest-neighbors employed in the predictions [4, 43, 55, 66]. Finally, data sparsity can also be tackled by acquiring new ratings with active learning techniques. In such techniques, the system interactively queries the user to gain a better understanding of his or her preferences. A more detailed description of active learning techniques can be found in Chap. 24 of this book.

These solutions, however, also have their own drawbacks. For instance, giving a default value to missing ratings may induce bias in the recommendations. Also, item content may not be available to compute ratings or similarities. This section presents two approaches proposed for the problems of limited coverage and sparsity: *graph-based* and *learning-based* and methods.

2.5.1 Graph-Based Methods

In graph-based approaches, the data is represented in the form of a graph where nodes are users, items or both, and edges encode the interactions or similarities between the users and items. For example, in Fig. 2.4, the data is modeled as a bipartite graph where the two sets of nodes represent users and items, and an edge connects user u to item i if there is a rating given to i by u in the system. A weight can also be given to this edge, such as the value of its corresponding rating. In another model, the nodes can represent either users or items, and an edge connects two nodes if the ratings corresponding two these nodes are sufficiently correlated. The weight of this edge can be the corresponding correlation value.

In these models, standard approaches based on correlation predict the rating of a user u for an item i using only the nodes directly connected to u or i. Graph-based

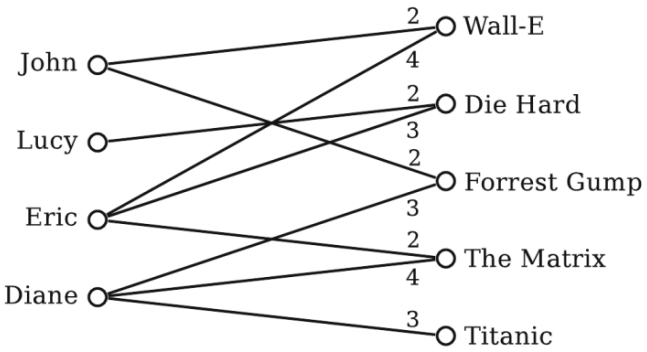

Fig. 2.4 A bipartite graph representation of the ratings of Fig. 2.1 (*only ratings with value in* $\{2, 3, 4\}$ *are shown*)

approaches, on the other hand, allow nodes that are not directly connected to influence each other by propagating information along the edges of the graph. The greater the weight of an edge, the more information is allowed to pass through it. Also, the influence of a node on another should be less if the two nodes are further away in the graph. These two properties, known as *propagation* and *attenuation* [22, 30], are often observed in graph-based similarity measures.

The transitive associations captured by graph-based methods can be used to recommend items in two different ways. In the first approach, the proximity of a user u to an item i in the graph is used directly to evaluate the relevance of i to u [16, 22, 30]. Following this idea, the items recommended to u by the system are those that are the "closest" to u in the graph. On the other hand, the second approach considers the proximity of two users or item nodes in the graph as a measure of similarity, and uses this similarity as the weights w_{uv} or w_{ij} of a neighborhood-based recommendation method [16, 45].

2.5.1.1 Path-Based Similarity

In path-based similarity, the distance between two nodes of the graph is evaluated as a function of the number of paths connecting the two nodes, as well as the length of these paths.

Let R be once again the $|U| \times |I|$ rating matrix, where r_{ui} is the rating given by user u to an item i. The adjacency matrix A of the user-item bipartite graph can be defined from R as

$$A = \begin{pmatrix} 0 & R^\top \\ R & 0 \end{pmatrix}.$$

The association between a user u and an item i can be defined as the sum of the weights of all distinctive paths connecting u to i (allowing nodes to appear more than once in the path), whose length is no more than a given maximum length K. Note that, since the graph is bipartite, K should be an odd number. In order to attenuate the contribution of longer paths, the weight given to a path of length k is defined as α^k, where $\alpha \in [0, 1]$. Using the fact that the number of length k paths between pairs of nodes is given by A^k, the user-item association matrix S_K is

$$\begin{aligned} S_K &= \sum_{k=1}^{K} \alpha^k A^k \\ &= (I - \alpha A)^{-1}(\alpha A - \alpha^K A^K). \end{aligned} \tag{2.29}$$

This method of computing distances between nodes in a graph is known as the *Katz* measure [35]. Note that this measure is closely related to the *Von Neumann Diffusion* kernel [17, 38, 41]

$$\begin{aligned} K_{\text{VND}} &= \sum_{k=0}^{\infty} \alpha^k A^k \\ &= (I - \alpha A)^{-1} \end{aligned} \tag{2.30}$$

and the *Exponential Diffusion* kernel

$$\begin{aligned} K_{\text{ED}} &= \sum_{k=0}^{\infty} \frac{1}{k!} \alpha^k A^k \\ &= \exp(\alpha A), \end{aligned} \tag{2.31}$$

where $A^0 = I$.

In recommender systems that have a large number of users and items, computing these association values may require extensive computational resources. In [30], spreading activation techniques are used to overcome these limitations. Essentially, such techniques work by first activating a selected subset of nodes as starting nodes, and then iteratively activating the nodes that can be reached directly from the nodes that are already active, until a convergence criterion is met.

Path-based methods, as well as the other graph-based approaches described in this section, focus on finding relevant associations between users and items, not predicting exact ratings. Therefore, such methods are better suited for item retrieval tasks, where explicit ratings are often unavailable and the goal is to obtain a short list of relevant items (i.e., the top-N recommendation problem).

memory than other types of approaches. Finally, since the parameters are usually learned offline, the online recommendation process is generally faster.

Learning-based methods that use neighborhood or similarity information can be divided in two categories: factorization methods and adaptive neighborhood learning methods. These categories are presented in the following sections.

2.5.2.1 Factorization Methods

Factorization methods [6, 7, 20, 40, 60, 68, 69] address the problems of limited coverage and sparsity by projecting users and items into a reduced latent space that captures their most salient features. Because users and items are compared in this dense subspace of high-level features, instead of the "rating space", more meaningful relations can be discovered. In particular, a relation between two users can be found, even though these users have rated different items. As a result, such methods are generally less sensitive to sparse data [6, 7, 60].

There are essentially two ways in which factorization can be used to improve recommender systems: (1) factorization of a sparse *similarity* matrix, and (2) factorization of a user-item *rating* matrix.

Factorizing the Similarity Matrix

Neighborhood similarity measures like the correlation similarity are usually very sparse since the average number of ratings per user is much less than the total number of items. A simple solution to densify a sparse similarity matrix is to compute a low-rank approximation of this matrix with a factorization method.

Let W be a symmetric matrix of rank n representing either user or item similarities. To simplify the presentation, we will suppose the latter case. We wish to approximate W with a matrix $\hat{W} = QQ^\top$ of lower rank $k < n$, by minimizing the following objective:

$$\begin{aligned} E(Q) &= ||W - QQ^\top||_F^2 \\ &= \sum_{i,j} \left(w_{ij} - \mathbf{q}_i \mathbf{q}_j^\top\right)^2, \end{aligned} \tag{2.33}$$

where $||M||_F = \sqrt{\sum_{i,j} m_{ij}^2}$ is the matrix Frobenius norm. Matrix $\hat{W}$ can be seen as a "compressed" and less sparse version of W. Finding the factor matrix Q is equivalent to computing the eigenvalue decomposition of W:

$$W = VDV^\top,$$

where D is a diagonal matrix containing the $|\mathcal{I}|$ eigenvalues of W, and V is a $|\mathcal{I}|\times|\mathcal{I}|$ orthogonal matrix containing the corresponding eigenvectors. Let V_k be a matrix formed by the k principal (normalized) eigenvectors of W, which correspond to the axes of the k-dimensional latent subspace. The coordinates $\mathbf{q}_i \in \mathbb{R}^k$ of an item i in this subspace is given by the ith row of matrix $Q = V_k D_k^{1/2}$. Furthermore, the item similarities computed in this latent subspace are given by matrix

$$\begin{aligned}\hat{W} &= QQ^\top \\ &= V_k D_k V_k^\top. \end{aligned} \tag{2.34}$$

This approach was used to recommend jokes in the Eigentaste system [20]. In Eigentaste, a matrix W containing the PC similarities between pairs of items is decomposed to obtain the latent subspace defined by the k principal eigenvectors of W. A user u, represented by the uth row $\mathbf{r}_u$ of the rating matrix R, is projected in the plane defined by V_k:

$$\mathbf{r}'_u \;=\; \mathbf{r}_u V_k.$$

In an offline step, the users of the system are clustered in this subspace using a recursive subdivision technique. Then, the rating of user u for an item i is evaluated as the mean rating for i made by users in the same cluster as u. This strategy is related to the well-known spectral clustering method [64].

Factorizing the Rating Matrix

The problems of cold-start and limited coverage can also be alleviated by factorizing the user-item rating matrix. Once more, we want to approximate the $|\mathcal{U}|\times|\mathcal{I}|$ rating matrix R of rank n by a matrix $\hat{R} = PQ^\top$ of rank $k < n$, where P is a $|\mathcal{U}|\times k$ matrix of *users* factors and Q a $|\mathcal{I}|\times k$ matrix of *item* factors. This task can be formulated as finding matrices P and Q which minimize the following function:

$$\begin{aligned} E(P,Q) \;&=\; ||R - PQ^\top||_F^2 \\ &=\; \sum_{u,i} \left(r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top\right)^2. \end{aligned} \tag{2.35}$$

The optimal solution can be obtained by the Singular Value Decomposition (SVD) of R: $P = U_k D_k^{1/2}$ and $Q = V_k D_k^{1/2}$, where D_k is a diagonal matrix containing the k largest singular values of R, and U_k, V_k respectively contain the left and right singular vectors corresponding to these values.

However, there is significant problem with applying SVD directly to the rating matrix R: most values r_{ui} of R are undefined, since there may not be a rating given to i by u. Although it is possible to assign a default value to r_{ui}, as mentioned above,

this would introduce a bias in the data. More importantly, this would make the large matrix R dense and, consequently, render impractical the SVD decomposition of R. A common solution to this problem is to learn the model parameters using only the known ratings [6, 40, 67, 69]. For instance, suppose the rating of user u for item i is estimated as

$$\hat{r}_{ui} = b_u + b_i + \mathbf{p}_u \mathbf{q}_i^\top, \tag{2.36}$$

where b_u and b_i are parameters representing the user and item rating biases. The model parameters can be learned by minimizing the following objective function:

$$E(P, Q, \mathbf{b}) = \sum_{r_{ui} \in \mathcal{R}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(||\mathbf{p}_u||^2 + ||\mathbf{q}_i||^2 + b_u^2 + b_i^2 \right). \tag{2.37}$$

The second term of the function is as a regularization term added to avoid overfitting. Parameter λ controls the level of regularization. A more comprehensive description of this recommendation approach can be found in Chap. 3 of this book.

The SVD model of Eq. (2.36) can be transformed into a similarity-based method by supposing that the profile of a user u is determined implicitly by the items he or she has rated. Thus, the factor vector of u can be defined as a weighted combination of the factor vectors $\mathbf{s}_j$ corresponding to the items j rated by this user:

$$\mathbf{p}_u = |\mathcal{I}_u|^{-\alpha} \sum_{j \in \mathcal{I}_u} c_{uj} \mathbf{s}_j. \tag{2.38}$$

In this formulation, α is a normalization constant typically set to $\alpha = 1/2$, and c_{uj} is a weight representing the contribution of item j to the profile of u. For instance, in the SVD++ model [40] this weight is defined as the bias corrected rating of u for item j: $c_{uj} = r_{ui} - b_u - b_j$. Other approaches, such as the FISM [34] and NSVD [53] models, instead use constant weights: $c_{uj} = 1$.

Using the formulation of Eq. (2.38), a rating r_{ui} is predicted as

$$\hat{r}_{ui} = b_u + b_i + |\mathcal{I}_u|^{-\alpha} \sum_{j \in \mathcal{I}_u} c_{uj} \mathbf{s}_j \mathbf{q}_i^\top. \tag{2.39}$$

Like the standard SVD model, the parameters of this model can be learned by minimizing the objective function of Eq. (2.37), for instance, using gradient descent optimization.

Note that, instead of having both user and item factors, we now have two different sets of item factors, i.e., $\mathbf{q}_i$ and $\mathbf{s}_j$. These vectors can be interpreted as the factors of an asymmetric item-item similarity matrix W, where

$$w_{ij} = \mathbf{s}_i \mathbf{q}_j^\top. \tag{2.40}$$

$$\begin{aligned} \underset{W,Q}{\text{minimize}} \quad & \frac{1}{2}\|R - RW\|_F^2 + \frac{\alpha}{2}\|F - FQ\|_F^2 + \frac{\beta_1}{2}\|W - Q\|_F^2 \\ & + \frac{\beta_2}{2}\left(\|W\|_F^2 + \|Q\|_F^2\right) + \lambda\left(\|W\|_1 + \|Q\|_1\right) \\ \text{subject to} \quad & W, Q \geq 0, \\ & \text{diag}(W) = 0, \ \text{diag}(Q) = 0. \end{aligned} \tag{2.45}$$

Parameter β_1 controls how much W and Q are allowed to be different from each other.

In [51], item reviews in the form of short texts were used as side information in the models described above. These models were shown to outperform the `SLIM` method without side information, as well as other approaches that use side information, in the top-N recommendation task.

2.6 Conclusion

One of the earliest approaches proposed for the task of item recommendation, neighborhood-based recommendation still ranks among the most popular methods for this problem. Although quite simple to describe and implement, this recommendation approach has several important advantages, including its ability to explain a recommendation with the list of the neighbors used, its computational and space efficiency which allows it to scale to large recommender systems, and its marked stability in an online setting where new users and items are constantly added. Another of its strengths is its potential to make serendipitous recommendations that can lead users to the discovery of unexpected, yet very interesting items.

In the implementation of a neighborhood-based approach, one has to make several important decisions. Perhaps the one having the greatest impact on the accuracy and efficiency of the recommender system is choosing between a user-based and an item-based neighborhood method. In typical commercial recommender systems, where the number of users far exceeds the number of available items, item-based approaches are typically preferred since they provide more accurate recommendations, while being more computationally efficient and requiring less frequent updates. On the other hand, user-based methods usually provide more original recommendations, which may lead users to a more satisfying experience. Moreover, the different components of a neighborhood-based method, which include the normalization of ratings, the computation of the similarity weights and the selection of the nearest-neighbors, can also have a significant influence on the quality of the recommender system. For each of these components, several different alternatives are available. Although the merit of each of these has been described in this document and in the literature, it is important to remember that the "best" approach may differ from one recommendation setting to the next. Thus, it is

important to evaluate them on data collected from the actual system, and in light of the particular needs of the application.

Finally, when the performance of a neighborhood-based approach suffers from the problems of limited coverage and sparsity, one may explore techniques based on dimensionality reduction or graphs. Dimensionality reduction provides a compact representation of users and items that captures their most significant features. An advantage of such approach is that it allows to obtain meaningful relations between pairs of users or items, even though these users have rated different items, or these items were rated by different users. On the other hand, graph-based techniques exploit the transitive relations in the data. These techniques also avoid the problems of sparsity and limited coverage by evaluating the relationship between users or items that are not "directly connected". However, unlike dimensionality reduction, graph-based methods also preserve some of the "local" relations in the data, which are useful in making serendipitous recommendations.

References

1. Adams, R.P., Dahl, G.E., Murray, I.: Incorporating side information into probabilistic matrix factorization using Gaussian processes. In: P. Grünwald, P. Spirtes (eds.) Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, pp. 1–9 (2010)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17**(6), 734–749 (2005)
3. Agarwal, D., Chen, B.C., Long, B.: Localized factor models for multi-context recommendation. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, pp. 609–617. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/2020408.2020504. URL http://doi.acm.org/10.1145/2020408.2020504
4. Balabanović, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM **40**(3), 66–72 (1997)
5. Baltrunas, L., Ricci, F.: Item weighting techniques for collaborative filtering. In: Knowledge Discovery Enhanced with Semantic and Social Information, pp. 109–126. Springer (2009)
6. Bell, R., Koren, Y., Volinsky, C.: Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: KDD '07: Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 95–104. ACM, New York, NY, USA (2007)
7. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: ICML '98: Proc. of the 15th Int. Conf. on Machine Learning, pp. 46–54. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
8. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Modeling and User-Adapted Interaction **10**(2–3), 147–180 (2000)
9. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
10. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proc. of the 14th Annual Conf. on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
11. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems **30**(1–7), 107–117 (1998)

The problem of extracting and using knowledge contained in Wikipedia was studied by several researchers [33, 46, 49]. Different techniques have been defined, which exploit the encyclopedic knowledge contained in Wikipedia for *selecting the most accurate semantic features* to represent the items, or for *generating new semantic features* to enrich the item representation.

The most prominent approaches which perform *feature selection* are Wikify! [33] and Tagme [46]. Wikify! allows to identify important concepts in a text representation by using keyword extraction, and then to link these concepts to the corresponding Wikipedia pages by exploiting WSD techniques. More specifically, Wikify! is a system for automatically cross-referencing documents with Wikipedia [85]. The system is trained on Wikipedia articles, and thus learns to disambiguate and detect links in the same way as Wikipedia editors [45].

Tagme [46] augments a text representation with pertinent hyperlinks to Wikipedia pages, by implementing an anchor disambiguation algorithm which exploits inter-relations between Wikipedia pages, as well as other heuristics. The main advantage of Tagme is its ability to annotate texts which are short and poorly composed, such as snippets coming from search engine result pages, tweets, news, etc.

An approach which leverages Wikipedia knowledge to *generate* new features for enriching items representation is Explicit Semantic Analysis (ESA) [49]. ESA provides a fine-grained semantic representation of text documents as a weighted vector of concepts derived from Wikipedia. Specifically, concepts correspond to Wikipedia articles, e.g. such as WOODY ALLEN, APPLE INC., or MACHINE LEARNING. Explicit Semantic Analysis resembles the well known Latent Semantic Analysis technique [35], whose representation is based on *latent* (and not comprehensible) features, rather than *explicit* (and comprehensible) concepts derived from Wikipedia (concepts explicitly defined and manipulated by humans).

In [48, 49], ESA was adopted for computing semantic relatedness of natural language texts, with better performance with respect to a keyword-based approach. In [43], ESA is adopted to enrich documents and queries to enhance traditional bag-of-words-based retrieval models, while in [8], ESA is used for enriching bag-of-words representing news or blog feeds before their clustering. ESA was also effectively used to augment the bag-of-words representation with Wikipedia-based features in the text categorization task [49].

Finally, the availability of Wikipedia knowledge in several languages and the multilingual alignment of Wikipedia articles allow to have cross-lingual and multilingual services. Potthast et al. [109] proposed a Wikipedia-based multilingual retrieval model for the analysis of cross-language similarity. They demonstrated that, given a query in a specific language, the most similar documents from a corpus in another language were properly ranked. They used Cross-Language Explicit Semantic Analysis (CL-ESA), an extension of ESA for cross-language retrieval. Recently, ESA was also used to develop the Cross-language Service Retriever tool (CroSeR), to support the cross-language linking of e-Government services to the Linked Open Data cloud [98].

4.3.2.1 Explicit Semantic Analysis

The idea behind ESA is to view an encyclopedia as a collection of concepts, each of which accompanied with a large body of text (the article content). The power of ESA is the capability of representing the Wikipedia knowledge base in a way that is directly used by machines, without the need for manually encoded common-sense knowledge. The gist of the technique is to use the high-dimensional space defined by these concepts in order to represent the meaning of natural language texts. ESA allows to leverage Wikipedia knowledge by defining relationships between terms and Wikipedia articles.

More formally, given a set of basic concepts $C = \{c_1, c_2, \ldots, c_n\}$, a term t is represented by a vector of weights $< w_1, w_2, \ldots, w_n >$, where w_i represents the strength of association between t and c_i. The set of concepts C are one to one associated to documents $D = \{d_1, d_2, \ldots, d_n\}$ (the Wikipedia articles). Hence, a sparse matrix T is built, called *ESA-matrix*, where each column corresponds to a concept (title of Wikipedia article), and each row corresponds to a term (word) that occurs in $\bigcup_{i=1\ldots n} d_i$. The entry $T[i, j]$ of the matrix represents the TF-IDF of term t_i in document d_j. Finally, length normalization is applied to each column to disregard differences in document length. This allows to define the semantics of a term t_i as a point in the n-dimensional semantic space of Wikipedia concepts. The weighed vector corresponding to a term t_i is called *semantic interpretation vector*. The semantics of a text fragment $< t_1, t_2, \ldots, t_k >$ (i.e. a sentence, a paragraph, an entire document) is obtained by computing the centroid (average vector) of the semantic interpretation vectors of the individual terms occurring in the fragment. This definition allows to partially perform WSD [49].

As an example consider the text fragment of a news title "Apple patents a Tablet Mac". Without deep knowledge of hi-tech industry and gadgets, one finds it hard to predict the content of the news item. Using Wikipedia it is possible to identify the following related concepts: APPLE COMPUTER (with the correct identification of the concept representing the computer company rather than the fruit), MAC OS, LAPTOP, AQUA (the GUI of Mac OS X), IPOD, and APPLE NEWTON (the name of Apple's early personal digital assistant).

4.3.2.2 CBRSs Leveraging Encyclopedic Knowledge

Even though the above mentioned indexing methods have been adopted for several tasks, they are not yet widely used in the context of learning user profiles and providing recommendations. However, CBRSs may benefit of the Wikipedia-based representation. Indeed, the feature generation process, adopted for example by ESA, can lead to richer item representations, able to improve the overlap between items and profiles. Indeed the new features allow to match items that did not share any keyword with the profile before the feature generation process. ESA is also able to introduce new related concepts for generating less obvious and more serendipitous (unexpected) recommendations.

In [91], an enhanced semantic TV-show representation for Personalized Electronic Program Guides is proposed. ESA is used to enrich the textual descriptions associated to TV shows with additional features extracted from Wikipedia, in order to improve the ranking of the most relevant items for each program genre. ESA is exploited to enrich a classic bag-of-words representation with 20, 40, or 60 new features, and it was adopted to enrich German TV-show descriptions. To this purpose, the German Wikipedia dump (released on October 13th, 2010 with a size of approximately 7.5 GB) was processed in order to obtain the corresponding German ESA-matrix. Results showed that the enhanced bag-of-words representation outperforms the classical bag-of-words one in terms of precision.

Besides the improvement of accuracy, the work carried out in [97] shows that, leveraging encyclopedic knowledge for representing user interests allows to introduce serendipitous topics and to obtain more understandable and transparent user profiles. Transparency is defined as the extent to which keywords in the user profile reflect the actual user interests. In that work user interests have been gathered from Facebook profiles by extracting both interests explicitly declared by users and those implicitly inferred from posts and other published content. The feature generation process implemented by ESA helps to introduce new serendipitous topics of interests, while the feature selection process implemented by Tagme helps to obtain more comprehensible user profiles, more representative of user interests.

These results are confirmed in the user study presented in [96], in which both ESA and Tagme are effectively used to improve the performance of a news recommender. News titles are extracted from a set of RSS feeds, and the profile of interests is built by extracting information from the Facebook and Twitter accounts of the user. The information extracted (news, posts, tweets) are represented using keywords, ESA concepts or Tagme concepts, respectively. The representation obtained by Tagme outperforms the others in terms of transparency and accuracy. This is probably due to the ability of Tagme to effectively annotate very short texts, such as news titles.

The ability of the ESA technique to cope with the cold-start problem is shown in [105], in which a CBRS in the context of non-fiction multimedia recommendation of TED lectures is presented. Using ESA as indexing method for titles and descriptions of talks allows to obtain the best performance with respect to other semantic representations, and this shows that a representation of items based on external knowledge is significantly more useful than the domain knowledge captured intrinsically by the other semantic methods.

4.3.2.3 BabelNet: An Encyclopedic Dictionary

Resources like Wikipedia lack full coverage for the lexicographic senses of lemmas, which is instead provided by a computational lexicon, such as WordNet . In this section we briefly describe a new resource, called BabelNet [100], which integrates the largest multilingual Web encyclopedia, i.e., Wikipedia, and the most popular

Index

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
DOI 10.1007/978-1-4899-7637-6