Chapter 1
Introduction to Recommender Systems Handbook

Francesco Ricci, Lior Rokach and Bracha Shapira

Abstract Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. In this introductory chapter we briefly discuss basic RS ideas and concepts. Our main goal is to delineate, in a coherent and structured way, the chapters included in this handbook and to help the reader navigate the extremely rich and detailed content that the handbook offers.

1.1 Introduction

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user [60, 85, 25]. 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. A 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 towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alter-

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

Lior Rokach
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Israel e-mail: liorrk@bgu.ac.il

Bracha Shapira
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Israel e-mail: bshapira@bgu.ac.il

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
DOI 10.1007/978-0-387-85820-3_1,

native items that a Web site, for example, may offer [85]. A case in point is a book recommender system that assists users to select a book to read. In the popular Web site, Amazon.com, the site employs a RS to personalize the online store for each customer [47]. Since recommendations are usually personalized, different users or user groups receive diverse 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 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 from users their preferences, which are either explicitly expressed, e.g., as ratings for products, or are inferred by interpreting user actions. For instance, a RS may consider the navigation to a particular product page as an implicit sign of preference for the items shown on that page.

RSs development initiated from a rather simple observation: individuals often rely on recommendations provided by others in making routine, daily decisions [60, 70]. 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 and which appear in the newspaper they read.

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

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

The explosive growth and variety of information available on the Web and the rapid introduction of new e-business services (buying products, product comparison, auction, 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, creating a sense that freedom may come to be regarded as a kind of misery-inducing tyranny [96].

RSs have proved in recent years to be a valuable means for coping with the information overload problem. Ultimately a RS addresses this phenomenon by pointing

a user towards new, not-yet-experienced items that may be relevant to the users 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. She may accept them or not and may provide, immediately or at a next stage, an implicit or explicit feedback. All these user actions and feedbacks can be stored in the recommender database and may be used for generating new recommendations in the next user-system interactions.

As noted above, the study of recommender systems is relatively new compared to research into 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 [35, 60, 70, 7]. In recent years, the interest in recommender systems has dramatically increased, as the following facts indicate:

1. Recommender systems play an important role in such highly rated Internet sites as Amazon.com, YouTube, Netflix, Yahoo, 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 movie rental service, awarded a million dollar prize to the team that first succeeded in improving substantially the performance of its recommender system [54].
2. There are dedicated conferences and workshops related to the field. We refer specifically to ACM Recommender Systems (RecSys), established in 2007 and now the premier annual event in recommender technology research and applications. In addition, sessions dedicated to RSs are frequently included in the more traditional conferences in the area of data bases, information systems and adaptive systems. Among these conferences are worth mentioning ACM SIGIR Special Interest Group on Information Retrieval (SIGIR), User Modeling, Adaptation and Personalization (UMAP), 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 recently a book introducing RSs techniques was published [48].
4. There have been several special issues in academic journals covering research and developments in the RS field. Among the journals that have dedicated issues to RS 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); and ACM Transactions on Information Systems (2004).

In this introductory chapter we briefly discuss basic RS ideas and concepts. Our main goal is not much to present a self-contained comprehensive introduction or 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 extremely rich and detailed content that the handbook offers.

The handbook is divided into five sections: techniques; applications and evaluation of RSs; interacting with RSs; RSs and communities; 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. It also deals with the practical aspects of designing recommender systems; describes design and implementation considerations; and sets guidelines for selecting the more suitable algorithms. The section also considers aspects that may affect RS design (domain, device, users, etc.). Finally, it discusses methods, challenges and measures to be applied in evaluating the developed systems.

The third section includes papers dealing with a number of issues related to how recommendations are presented, browsed, explained and visualized. The techniques that make the recommendation process more structured and conversational are discussed here.

The fourth section is fully dedicated to a rather new topic, exploiting user-generated content (UGC) of various types (tags, search queries, trust evaluations, etc.) to generate innovative types of recommendations and more credible ones. Despite its relative newness, this topic is essentially rooted in the core idea of a collaborative recommender,

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

1.2 Recommender Systems Function

In the previous section we defined RSs as software tools and techniques providing users with suggestions for items a user may wish to utilize. Now we want to refine this definition illustrating a range of possible roles that a RS can play. First of all, 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 (e.g., Expedia.com) or a destination management organization (e.g., Visitfinland.com) to increase its turnover (Expedia), i.e., sell more hotel rooms, or to increase the number of tourists to the destination [86]. Whereas, the user's primary motivations for accessing the two systems is to find a suitable hotel and interesting events/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 a 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 a RS is to enable the user to select items that might be hard to find without a precise recommendation. For instance, in a movie RS such as Netflix, the service provider is interested in renting all the DVDs in the catalogue, not just the most popular ones. This could be difficult without a RS since the service provider cannot afford the risk of advertising movies that are not likely to suit a particular user's taste. Therefore, a RS suggests or advertises unpopular movies 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, she will also enjoy using the system. The combination of effective, i.e., 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 Web site which, when visited, recognizes the old customer and treats him as a valuable visitor. This is a normal feature of a RS since many RSs compute recommendations, leveraging the information acquired from the user in previous interactions, e.g., her ratings of items. Consequently, the longer the user interacts with the site, the more refined her user model becomes, i.e., the system representation of the user's preferences, and the more the recommender output can be effectively customized to match the user's preferences.
- *Better understand what the user wants.* Another important function of a RS, which can be leveraged to many other applications, is the description of the user's preferences, either collected explicitly or predicted by the system. The service provider may then decide to re-use 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

[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.

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 a RS, if it will effectively support their tasks or goals. Consequently a RS must balance the needs of these two players and offer a service that is valuable to both.

Herlocker et al. [25], in a paper that has become a classical reference in this field, define eleven popular tasks that a RS can assist in implementing. Some may be considered as the main or core tasks that are normally associated with a RS, i.e., to offer suggestions for items that may be useful to a user. Others might be considered as more "opportunistic" ways to exploit a 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 Web page (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 one- to five-star scale). This is the main recommendation task that many commercial systems address (see, for instance, Chapter 9). 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 (Chapter 18 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 [99], [39].
- *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 [87].

- *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 been also supported by adaptive hypermedia techniques [23].
- *Find credible recommender:* Some users do not trust recommender systems thus they play with them to see how good they are in making recommendations. Hence, some 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 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 him with 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 it 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 a leverage for holding the user tightly to the application (as we mentioned above 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 a car RS, a user, who has already bought her new car is aware that the rating entered in the system is more likely to be useful for other users rather than for the next time she will buy a car.
- *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 some malicious users that may use the system just to promote or penalize certain items (see Chapter 25).

As these various points indicate, the role of a RS within an information system can be quite diverse. This diversity calls for the exploitation of a range of different knowledge sources and techniques and in the next two sections we discuss the data a 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 they can be exploited or not depends on the recommendation

technique (see also section 1.4). This will become clearer in the various chapters included in this handbook (see in particular Chapter 11).

In general, there are recommendation techniques that are knowledge poor, i.e., they use very simple and basic data, such as user ratings/evaluations for items (Chapters 5, 4). Other techniques are much more knowledge dependent, e.g., using ontological descriptions of the users or the items (Chapter 3), or constraints (Chapter 6), or social relations and activities of the users (Chapter 19). In any case, as a general classification, data used by RSs refers to three kinds of objects: items, users, and transactions, i.e., relations between users and 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 for the user, or negative if the item is not appropriate and the user made a wrong decision when selecting it. We note that when a user is acquiring an item she 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 to searching and reading news items. If a selected item is relevant for the user this cost is dominated by the benefit of having acquired a 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, Web pages, books, CDs, 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, travels, jobs [72].

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 (such as 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, but even as a concept in an ontological representation of the domain (Chapter 3).

Users. Users of a 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 some items. In a demographic RS, socio-demographic attributes such as age, gender, profession, and education, are used. User data is said to constitute the user model [21, 32]. 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, a RS can be viewed as a tool that generates recommendations by building and exploiting user models [19, 20]. Since no personalization is possible without a convenient user model, unless the recommendation is non-personalized, as in the top-10 selection, the user model will always play a central role. For instance, considering again a collaborative filtering approach, the user is either profiled directly by its ratings to items or, using these ratings, the system derives a vector of factor values, where users differ in how each factor weights in their model (Chapters 5 and 4).

Users can also be described by their behavior pattern data, for example, site browsing patterns (in a Web-based recommender system) [107], or travel search patterns (in a travel recommender system) [60]. Moreover, user data may include relations between users such as the trust level of these relations between users (Chapter 20). A 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 an explicit feedback the user has provided, such as the rating for the selected item.

In fact, ratings are the most popular form of transaction data that a RS collects. These ratings may be collected explicitly or implicitly. In the explicit collection of ratings, the user is asked to provide her opinion about an item on a rating scale. According to [93], 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 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 she purchased the item somewhere else).

Another form of user evaluation consists of tags associated by the user with the items the system presents. For instance, in Movielens RS (http://movielens.umn.edu) tags represent how MovieLens users feel about a movie, e.g.: "too long", or "acting". Chapter 19 focuses on these types of transactions.

In transactions collecting implicit ratings, the system aims to infer the users opinion based on the user's actions. For example, if a user enters the keyword "Yoga" at

Amazon.com she will be provided with a long list of books. 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 Chapter 13). 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 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 [60].

1.4 Recommendation Techniques

In order to implement its core function, identifying the 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 of them, or at least compare the utility of some items, and then decide what 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 a 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 a RS, consider, for instance, a simple, non-personalized, recommendation algorithm that recommends just the most popular songs. The rationale for using this approach is that in absence of more precise information about the user's preferences, a popular song, i.e., something that is liked (high utility) by many users, will also be probably liked by a generic user, at least more than another randomly selected song. Hence the utility of these 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 [3]. They 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, i.e., 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, i.e., 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 hypothe-

size that an item is 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 kind of knowledge about users, items, and the utility function itself (see section 1.3) [25]. 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 (possibly none) about the user who is requesting the recommendation, 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 [25].

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" [1]. For instance, the utility of an item for a user can be influenced by the domain knowledge of the user (e.g., expert vs. beginning users of a digital camera), or can depend on the time when the recommendation is requested. Or the user may be more interested in items (e.g., a restaurant) closer to his current location. Consequently, the recommendations must be adapted to these specific additional details and as a result it becomes harder and harder 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, the knowledge used, but especially in regard to the recommendation algorithm, i.e., how the prediction of the utility of a recommendation, as mentioned at the beginning of this section, is made. Other differences relate to how the recommendations are finally assembled and presented to the user in response to user requests. These aspects are also discussed later in this introduction.

To provide a first overview of the different types of RSs, we want to quote a taxonomy provided by [25] that has become a classical way of distinguishing between recommender systems and referring to them. [25] 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. Chapter 3 provides an overview of content-based recommender systems, imposing some order among the extensive and diverse aspects involved in their design and implementation. It presents the basic concepts and terminology of content-based RSs, their high level architecture, and their main advantages and drawbacks. The chapter then surveys state-of-the-art systems that have been adopted in several application domains. The survey encompasses a thorough description of both classical and advanced techniques for representing items and user profiles. Finally, it discusses trends and future research which might lead towards the next generation of recommender systems.

Collaborative filtering: The simplest and original implementation of this approach [93] recommends to the active user the 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 [94] 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 4 presents a comprehensive survey of neighborhood-based methods for collaborative filtering. Neighborhood 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. Nearest-neighbors methods enjoy considerable popularity due to their simplicity, efficiency, and their ability to produce accurate and personalized recommendations. The authors will address the essential decisions that are required when implementing a neighborhood-based recommender system and provide practical information on how to make such decisions.

Finally, the chapter deals with problems of data sparsity and limited coverage, often observed in large commercial recommender systems. A few solutions to overcome these problems are presented.

Chapter 5 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. The assumption is that different recommendations should be generated for different demographic niches. Many Web sites adopt simple and effective personalization solutions based on demographics. For example, users are dispatched to particular Web sites 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 into demographic systems [59].

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 [22, 87]. In these systems a similarity function estimates how much the user 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.

Constraint-based systems are another type of knowledge-based RSs (Chapter 6). In terms of used knowledge, both systems are similar: user requirements are col-

lected; 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.

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).

Community-based: This type of system recommends items based on the preferences of the users friends. This technique follows the epigram "Tell me who your friends are, and I will tell you who you are". [8, 14]. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals [103]. This observation, combined with the growing popularity of open social networks, is generating a rising interest in community-based systems or, as or as they usually referred to, social recommender systems [34]. This type of RSs 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.

The research in this area is still in its early phase and results about the systems performance are mixed. For example, [34, 64] report that overall, social-network-based recommendations are no more accurate than those derived from traditional CF approaches, except in special cases, such as when user ratings of a specific item are highly varied (i.e. controversial items) or for cold-start situations, i.e., where the users did not provide enough ratings to compute similarity to other users. Others have showed that in some cases social-network data yields better recommendations than profile similarity data [37] and that adding social network data to traditional CF improves recommendation results [36]. The chapter 20 provides a survey of the findings in this field and analyzes current results.

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, i.e., 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 [25] for the precise descriptions).

As we have already mentioned, the context of the user when 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. Or a restaurant recommendation for a

Saturday evening with your friends should be different from that suggested for a workday lunch with co-workers.

Chapter 7 presents the general notion of context and how it can be modeled in RSs. Discussing 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 different algorithmic paradigms for incorporating contextual information into the recommendation process are discussed: reduction-based (pre-filtering), contextual post filtering, and context modeling. In reduction-based (pre-filtering) 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.

1.5 Application and Evaluation

Recommender system research is being conducted with a strong emphasis on practice and commercial applications, since, aside from its theoretical contribution, is generally aimed at practically improving commercial RSs. Thus, RS research involves practical aspects that apply to the implementation of these systems. These aspects are relevant to different stages in the life cycle of a RS, namely, the design of the system, its implementation and its maintenance and enhancement during system operation.

The aspects that apply to the design stage include factors that might affect the choice of the algorithm. The first factor to consider, the application's domain, has a major effect on the algorithmic approach that should be taken. [72] 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, and IPTV.
- Content - personalized newspapers, recommendation for documents, recommendations of Web pages, e-learning applications, and e-mail filters.
- E-commerce - recommendations for consumers of products to buy such as books, cameras, PCs etc.
- Services - recommendations of travel services, recommendation of experts for consultation, recommendation of houses to rent, or matchmaking services.

As recommender systems become more popular, interest is aroused in the potential advantages in new applications, such as recommending friends or tweets to

follow as in www.tweeter.com. Hence, 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 a 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 one could be able to select the optimal recommender algorithm and to design an effective human-computer interaction.

Chapter 11 of this handbook provides guidelines for matching the application domain to the recommendation technique. Burke and Ramezani in their chapter provide a new classification of recommender systems. Unlike former classifications of RSs (such as [25, 94, 3, 7]), Burke and Ramezani take an AI-centric approach, and focus on the knowledge sources required for different recommendation approaches, and the constraints related to them as a primer guideline to choosing the algorithm. The chapter discusses the applicability of various recommendation techniques for different types of problems and suggests decision-making guidelines in selecting these techniques.

The chapter explicitly aims at system implementers as "recommenders" for the right recommendation approach. The authors describe the knowledge sources that are available to a recommender systems in different domains and identify what knowledge sources are required for each recommendation technique. This implies that the design of a recommender system should first emphasize the analysis of the available sources of knowledge, and then decide about the algorithm accordingly.

Another example of the need to adjust the recommender approach to the domain is described in Chapter 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. 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. The chapter presents the particular requirements of RSs for TEL; the user tasks that are supported in TEL settings; and how these tasks compare to typical user tasks in other RSs. For example, one particular user task for TEL –"find novel resources"– attempts to recommend only new or novel items. Or, to cite another example, – "find new pathways" – is concerned with recommending alternative pathways through the learning resources. The chapter presents an analysis of the filtering approaches that could be useful for TEL along with a survey of existing TEL systems illustrating the recommendation techniques that have been deployed in these systems.

Chapter 10 discusses practical aspects of RS development and aims at providing practical guidelines to the design, implementation and evaluation of personalized systems. Besides the prediction algorithm, many other factors need to be considered when designing a RS. Chapter 10 lists some of these elements: the type of target users and their context; the devices that they would use; the role of the recommendation within the application; the goal of the recommendation; and, as mentioned previously, the data that is available.

The authors propose to build a model of the environment based on three dimensions: system users; the characteristics of the data; and the overall application. The recommender system design will be based on this model. The authors illustrate their guidelines and the model on a news recommendation system that they have developed.

Another important issue related to the practical side of RS deployment is the necessity of evaluating them. Evaluation is required at different stages of the systems life cycle for various purposes [25, 1]. 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 are compared with user interactions. The off-line evaluation consists of running several algorithms on the same datasets of user interactions (e.g., ratings) and comparing their performance. 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 [11] in order to ensure reliable results.

Evaluation is also required after the system has been launched. 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, e.g., because the performance of the system is not as expected. At this stage it is usually beneficial to perform on-line evaluation with real users of the system and analyze system logs in order to enhance system performance. In addition, most of the algorithms include parameters, such as weights thresholds, the number of neighbors, etc., requiring constant adjustment and calibration.

Another type of evaluation is a focused user study that can be 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 users 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.

Evaluation is also discussed in Chapter 12 in the context of TEL systems. The authors provide a detailed analysis of the evaluation methods and tools that can be employed for evaluating TEL recommendation techniques against a set of criteria that are proposed for each of the selected components (e.g., user model, domain model, recommendation strategy and algorithm).

Chapter 8 details three types of experiments that can be conducted in order to evaluate recommender systems. It presents their advantages and disadvantages, and defines guidelines for choosing the methods for evaluation them. Unlike existing discussions of evaluation in the literature that usually speaks about the accuracy of an algorithms prediction [25] 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 systems 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.

When discussing the practical aspects of RSs, it may be beneficial to analyze real system implementations. The idea is to test theoretically intuitive assumptions in order to determine if they work in practice. The major problem that one must face in this case comes from the fact that the owners of commercial RSs are generally unwilling to reveal their practices and there are only relatively few opportunities for such cooperation.

Chapter 9 reports on such an opportunity and describes the operation of a real RS, illustrating the practical aspects that apply to the implementation stage of the RS development and its evaluation. This description focuses on the integration of a RS into the production environment of Fastweb, one of the largest European IP Television (IPTV) providers. The chapter describes the requirements and considerations, including scaling and accuracy, that led to the choice of the recommender algorithms. It also describes the off-line and on-line evaluations that took place and illustrates how the system is adjusted accordingly.

1.6 Recommender Systems and Human Computer Interaction

As we have illustrated in previous sections, researchers have chiefly been concerned with designing a range of technical solutions, leveraging various sources of knowledge to achieve better predictions about what is liked and how much by the target user. The underlying assumption behind this research activity is that just presenting these correct recommendations, i.e., the best options, should be enough. 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 it is not so easy to deliver recommendations.

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 [105, 30, 24, 97, 33].

[105] was among the first to point out that the effectiveness of a 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) 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. [105] found that from a users perspective, an effective recommender system must inspire trust in the system; it must have a system logic that is at least somewhat transparent; it should point users towards new, not-yet-experienced items; it should provide details