Chapter 1

An Introduction to Recommender Systems

"*Many receive advice, only the wise profit from it.*" – Harper Lee

1.1 Introduction

The increasing importance of the Web as a medium for electronic and business transactions has served as a driving force for the development of recommender systems technology. An important catalyst in this regard is the ease with which the Web enables users to provide feedback about their likes or dislikes. For example, consider a scenario of a content provider such as Netflix. In such cases, users are able to easily provide feedback with a simple click of a mouse. A typical methodology to provide feedback is in the form of *ratings*, in which users select numerical values from a specific evaluation system (e.g., five-star rating system) that specify their likes and dislikes of various items.

Other forms of feedback are not quite as explicit but are even easier to collect in the Web-centric paradigm. For example, the simple act of a user buying or browsing an item may be viewed as an endorsement for that item. Such forms of feedback are commonly used by online merchants such as Amazon.com, and the collection of this type of data is completely effortless in terms of the work required of a customer. The basic idea of recommender systems is to utilize these various sources of data to infer customer interests. The entity to which the recommendation is provided is referred to as the *user*, and the product being recommended is also referred to as an *item*. Therefore, recommendation analysis is often based on the previous interaction between users and items, because past interests and proclivities are often good indicators of future choices. A notable exception

C.C. Aggarwal, *Recommender Systems: The Textbook*,
DOI 10.1007/978-3-319-29659-3_1

is the case of *knowledge-based recommender systems*, in which the recommendations are suggested on the basis of user-specified *requirements* rather than the past history of the user.

So, what is the basic principle that underlies the working of recommendation algorithms? The basic principle of recommendations is that significant dependencies exist between user- and item-centric activity. For example, a user who is interested in a historical documentary is more likely to be interested in another historical documentary or an educational program, rather than in an action movie. In many cases, various categories of items may show significant correlations, which can be leveraged to make more accurate recommendations. Alternatively, the dependencies may be present at the finer granularity of individual items rather than categories. These dependencies can be *learned* in a data-driven manner from the ratings matrix, and the resulting model is used to make predictions for target users. The larger the number of rated items that are available for a user, the easier it is to make robust predictions about the future behavior of the user. Many different learning models can be used to accomplish this task. For example, the collective buying or rating behavior of various users can be leveraged to create cohorts of similar users that are interested in similar products. The interests and actions of these cohorts can be leveraged to make recommendations to individual members of these cohorts.

The aforementioned description is based on a very simple family of recommendation algorithms, referred to as *neighborhood models*. This family belongs to a broader class of models, referred to as *collaborative filtering*. The term “collaborative filtering” refers to the use of ratings from multiple users in a collaborative way to predict missing ratings. In practice, recommender systems can be more complex and data-rich, with a wide variety of auxiliary data types. For example, in content-based recommender systems, the content plays a primary role in the recommendation process, in which the ratings of users and the attribute descriptions of items are leveraged in order to make predictions. The basic idea is that user interests can be modeled on the basis of properties (or *attributes*) of the items they have rated or accessed in the past. A different framework is that of *knowledge-based systems*, in which users interactively specify their interests, and the user specification is combined with domain knowledge to provide recommendations. In advanced models, contextual data, such as temporal information, external knowledge, location information, social information, or network information, may be used.

This book will study all types of basic systems, including collaborative, content-based, and knowledge-based systems. We will also discuss both the basic and the enhanced models of recommender systems in different domains. We will study various aspects of the robustness of recommender systems, such as attack models, and the construction of trustworthy models. In addition, a variety of evaluation and hybridization models for recommender systems will be studied thoroughly. In this chapter, the goal is to provide an overview of the wide diversity of work in the field of recommender systems, and also relate the various topics to the individual chapters of this book.

This chapter is organized as follows. Section 1.2 discusses the main goals of recommender systems. Section 1.3 will introduce the basic models and evaluation methods used in recommender systems. The use of recommender systems in various data domains is discussed in section 1.4. Advanced models for recommender systems are discussed in section 1.5. Section 1.6 discusses the conclusions and summary.

1.2 Goals of Recommender Systems

Before discussing the goals of recommender systems, we introduce the various ways in which the recommendation problem may be formulated. The two primary models are as follows:

1. *Prediction version of problem:* The first approach is to predict the rating value for a user-item combination. It is assumed that *training* data is available, indicating user preferences for items. For m users and n items, this corresponds to an incomplete $m \times n$ matrix, where the specified (or *observed*) values are used for training. The missing (or *unobserved*) values are predicted using this training model. This problem is also referred to as the *matrix completion problem* because we have an incompletely specified matrix of values, and the remaining values are predicted by the learning algorithm.

2. *Ranking version of problem:* In practice, it is not necessary to predict the ratings of users for specific items in order to make recommendations to users. Rather, a merchant may wish to recommend the top-k items for a particular user, or determine the top-k users to target for a particular item. The determination of the top-k items is more common than the determination of top-k users, although the methods in the two cases are exactly analogous. Throughout this book, we will discuss only the determination of the top-k items, because it is the more common setting. This problem is also referred to as the *top-k recommendation problem*, and it is the ranking formulation of the recommendation problem.

In the second case, the absolute values of the predicted ratings are not important. The first formulation is more general, because the solutions to the second case can be derived by solving the first formulation for various user-item combinations and then ranking the predictions. However, in many cases, it is easier and more natural to design methods for solving the ranking version of the problem directly. Such methods will be discussed in Chapter 13.

Increasing product sales is the primary goal of a recommender system. Recommender systems are, after all, utilized by merchants to increase their profit. By recommending carefully selected items to users, recommender systems bring relevant items to the attention of users. This increases the sales volume and profits for the merchant. Although the primary goal of a recommendation system is to increase revenue for the merchant, this is often achieved in ways that are less obvious than might seem at first sight. In order to achieve the broader *business-centric* goal of increasing revenue, the common *operational* and *technical* goals of recommender systems are as follows:

1. *Relevance:* The most obvious operational goal of a recommender system is to recommend items that are relevant to the user at hand. Users are more likely to consume items they find interesting. Although relevance is the primary operational goal of a recommender system, it is not sufficient in isolation. Therefore, we discuss several secondary goals below, which are not quite as important as relevance but are nevertheless important enough to have a significant impact.

2. *Novelty:* Recommender systems are truly helpful when the recommended item is something that the user has not seen in the past. For example, popular movies of a preferred genre would rarely be novel to the user. Repeated recommendation of popular items can also lead to reduction in sales diversity [203].

3. *Serendipity:* A related notion is that of *serendipity* [229], wherein the items recommended are somewhat unexpected, and therefore there is a modest element of lucky

discovery, as opposed to obvious recommendations. Serendipity is different from novelty in that the recommendations are truly *surprising* to the user, rather than simply something they did not know about before. It may often be the case that a particular user may only be consuming items of a specific type, although a latent interest in items of other types may exist which the user might themselves find surprising. Unlike novelty, serendipitous methods focus on discovering such recommendations.

For example, if a new Indian restaurant opens in a neighborhood, then the recommendation of that restaurant to a user who normally eats Indian food is novel but not necessarily serendipitous. On the other hand, when the same user is recommended Ethiopian food, and it was unknown to the user that such food might appeal to her, then the recommendation is serendipitous. Serendipity has the beneficial side effect of increasing sales diversity or beginning a new trend of interest in the user. Increasing serendipity often has long-term and strategic benefits to the merchant because of the possibility of discovering entirely new areas of interest. On the other hand, algorithms that provide serendipitous recommendations often tend to recommend irrelevant items. In many cases, the longer term and strategic benefits of serendipitous methods outweigh these short-term disadvantages.

4. *Increasing recommendation diversity:* Recommender systems typically suggest a list of top-k items. When all these recommended items are very similar, it increases the risk that the user might not like *any* of these items. On the other hand, when the recommended list contains items of different types, there is a greater chance that the user might like at least one of these items. Diversity has the benefit of ensuring that the user does not get bored by repeated recommendation of similar items.

Aside from these concrete goals, a number of soft goals are also met by the recommendation process both from the perspective of the user and merchant. From the perspective of the user, recommendations can help improve overall user satisfaction with the Web site. For example, a user who repeatedly receives relevant recommendations from Amazon.com will be more satisfied with the experience and is more likely to use the site again. This can improve user loyalty and further increase the sales at the site. At the merchant end, the recommendation process can provide insights into the needs of the user and help customize the user experience further. Finally, providing the user an explanation for why a particular item is recommended is often useful. For example, in the case of Netflix, recommendations are provided along with previously watched movies. As we will see later, some recommendation algorithms are better suited to providing explanations than others.

There is a wide diversity in the types of products recommended by such systems. Some recommender systems, such as Facebook, do not directly recommend products. Rather they may recommend social connections, which have an indirect benefit to the site by increasing its usability and advertising profits. In order to understand the nature of these goals, we will discuss some popular examples of historical and current recommender systems. These examples will also showcase the broad diversity of recommender systems that were built either as research prototypes, or are available today as commercial systems in various problem settings.

GroupLens Recommender System

GroupLens was a pioneering recommender system, which was built as a research prototype for recommendation of Usenet news. The system collected ratings from Usenet readers and used them to predict whether or not other readers would like an article before they read it.

Some of the earliest automated collaborative filtering algorithms were developed in the GroupLens[1] setting. The general ideas developed by this group were also extended to other product settings such as books and movies. The corresponding recommender systems were referred to as *BookLens* and *MovieLens*, respectively. Aside from its pioneering contributions to collaborative filtering research, the GroupLens research team was notable for releasing several data sets during the early years of this field, when data sets were not easily available for benchmarking. Prominent examples include three data sets [688] from the MovieLens recommender system. These data sets are of successively increasing size, and they contain 10^5, 10^6, and 10^7 ratings, respectively.

Amazon.com Recommender System

Amazon.com [698] was also one of the pioneers in recommender systems, especially in the commercial setting. During the early years, it was one of the few retailers that had the foresight to realize the usefulness of this technology. Originally founded as a book e-retailer, the business expanded to virtually all forms of products. Consequently, Amazon.com now sells virtually all categories of products such as books, CDs, software, electronics, and so on. The recommendations in Amazon.com are provided on the basis of explicitly provided ratings, buying behavior, and browsing behavior. The ratings in Amazon.com are specified on a 5-point scale, with lowest rating being 1-star, and the highest rating being 5-star. The customer-specific buying and browsing data can be easily collected when users are logged in with an account authentication mechanism supported by Amazon. Recommendations are also provided to users on the main Web page of the site, whenever they log into their accounts. In many cases, explanations for recommendations are provided. For example, the relationship of a recommended item to previously purchased items may be included in the recommender system interface.

The purchase or browsing behavior of a user can be viewed as a type of *implicit rating*, as opposed to an *explicit rating*, which is specified by the user. Many commercial systems allow the flexibility of providing recommendations both on the basis of explicit and implicit feedback. In fact, several models have been designed (cf. section 3.6.4.6 of Chapter 3) to jointly account for explicit and implicit feedback in the recommendation process. Some of the algorithms used by early versions of the Amazon.com recommender system are discussed in [360].

Netflix Movie Recommender System

Netflix was founded as a mail-order digital video disc (DVD) rental company [690] of movies and television shows, which was eventually expanded to streaming delivery. At the present time, the primary business of Netflix is that of providing streaming delivery of movies and television shows on a subscription basis. Netflix provides users the ability to rate the movies and television shows on a 5-point scale. Furthermore, the user actions in terms of watching various items are also stored by Netflix. These ratings and actions are then used by Netflix to make recommendations. Netflix does an excellent job of providing *explanations* for the recommended items. It explicitly provides examples of recommendations based on specific items that were watched by the user. Such information provides the user with additional

[1] The term "GroupLens" currently refers to the academic group at the University of Minnesota [687] that developed these algorithms. This group continues to work in the area of recommender systems, and has made many pioneering contributions over the years.

information to decide whether or not to watch a specific movie. Presenting meaningful explanations is important to provide the user with an understanding of *why* they might find a particular movie interesting. This approach also makes it more likely for the user to act on the recommendation and truly improves the user experience. This type of interesting approach can also help improve customer loyalty and retention.

Netflix has contributed significantly to the research community as a result of the *Netflix Prize contest*. This contest was designed to provide a forum for competition among various collaborative filtering algorithms contributed by contestants. A data set of Netflix movie ratings was released, and the task was to predict ratings of particular user-item combinations. For this purpose, Netflix provided both a *training* data set, and a *qualifying* data set. The training data set contained 100,480,507 ratings that 480,189 users gave to 17,770 movies. The training set included a smaller *probe set* containing 1,408,395 ratings. The probe set was based on more recent ratings than the remaining training data, and it was statistically similar to the portion of the data set with hidden ratings. This portion of the data set was referred to as the *qualifying* data set, and it contained over 2,817,131 triplets of the form $\langle User, Movie, GradeDate\rangle$. Note that the triplet did not contain the actual rating, which was known only to the judges. Users needed to predict the ratings in the qualifying data set based on models of the training data. This prediction was scored by the judges (or an equivalent automated system), and the users were (continuously) informed of the prediction results on only half the qualifying data set on the *leader-board*. This half of the qualifying data set was referred to as the *quiz set*. The remaining half was used as the *test set* for computing the final score and determining the prize-winners. The scores of the remaining half were never revealed to the users until the very end. Furthermore, it was not revealed to the contestants which of the triplets in the qualifying set belonged to the quiz set, and which belonged to the test set. The reason for this unusual arrangement on the test set was to ensure that the users did not leverage the scores on the leader-board to overfit their algorithms to the test set. Issues related to overfitting will be described in Chapter 7 on evaluation algorithms. Indeed, Netflix's framework for handling the contestant entries is an excellent example of proper evaluation design of recommendation algorithms.

The probe set, quiz set, and test set were designed to have similar statistical characteristics. Prizes were given based on improvement of Netflix's own recommendation algorithm, known as *Cinematch*, or by improvement of the previous best score by a certain threshold. Many well-known recommendation algorithms, such as latent factor models, were popularized by the Netflix contest. The Netflix Prize contest is notable for its numerous contributions to recommendation [71, 373] research.

Google News Personalization System

The Google News personalization system [697] is able to recommend news to users based on their history of clicks. The clicks are associated with specific users based on identification mechanisms enabled by Gmail accounts. In this case, news articles are treated as items. The act of a user clicking on a news article can be viewed as a positive rating for that article. Such ratings can be viewed as *unary ratings*, in which a mechanism exists for a user to express their affinity for an item, but no mechanism exists for them to show their dislike. Furthermore, the ratings are *implicit*, because they are *inferred* from user actions rather than being explicitly specified by the user. Nevertheless, variations of the approach can also be applied to cases where ratings are explicitly specified. Collaborative recommendation algorithms are applied to the collected ratings, so that inferences can be made about the

Table 1.1: Examples of products recommended by various real-world recommender systems

System	Product Goal
Amazon.com [698]	Books and other products
Netflix [690]	DVDs, Streaming Video
Jester [689]	Jokes
GroupLens [687]	News
MovieLens [688]	Movies
last.fm [692]	Music
Google News [697]	News
Google Search [696]	Advertisements
Facebook [691]	Friends, Advertisements
Pandora [693]	Music
YouTube [694]	Online videos
Tripadvisor [695]	Travel products
IMDb [699]	Movies

personalized articles for specific users. A description of a collaborative filtering system for Google News is provided in [175]. More details of the Google News personalization engine are discussed in section 13.8.1.2 of Chapter 13.

Facebook Friend Recommendations

Social networking sites often recommend potential friends to users in order to increase the number of social connections at the site. Facebook [691] is one such example of a social networking Web site. This kind of recommendation has slightly different goals than a product recommendation. While a product recommendation directly increases the profit of the merchant by facilitating product sales, an increase in the number of social connections improves the experience of a user at a social network. This, in turn, encourages the growth of the social network. Social networks are heavily dependent on the growth of the network to increase their advertising revenues. Therefore, the recommendation of potential friends (or *links*) enables better growth and connectivity of the network. This problem is also referred to as *link prediction* in the field of social network analysis. Such forms of recommendations are based on *structural relationships* rather than ratings data. Therefore, the nature of the underlying algorithms is completely different. The link recommendation problem is explored in detail in Chapter 10. The relationship of computational advertising to recommender system technology is discussed in Chapter 13.

1.2.1 The Spectrum of Recommendation Applications

In the following, we will provide a brief overview of the application-specific goals accomplished by various implementations of recommender systems. A brief overview of the products suggested and the goals accomplished by various recommender systems are illustrated in Table 1.1. Many of these recommender systems are focused on traditional e-commerce applications for various products, including books, movies, videos, travel, and other goods and services. The broader applicability of recommender systems to e-commerce applications is discussed in [530]. However, recommender systems have expanded beyond the traditional domain of product recommendations. It is noteworthy that some of the systems in Table 1.1

may not recommend specific products. An example is the Google search application, which may advertise products along with their search results. This is the area of *computational advertising*, which is a distinct area in its own right, but it is nevertheless closely related to recommender systems. This area is discussed in detail in section 13.8.2 of Chapter 13. Similarly, Facebook recommends friends, and online recruitment sites recommend employers and job-seekers to one another. The last of these systems is also referred to as a *reciprocal recommender*. The models for some of these recommendation algorithms are quite different from those of traditional recommender systems. This book will study many of these variations in detail.

1.3 Basic Models of Recommender Systems

The basic models for recommender systems work with two kinds of data, which are (i) the user-item interactions, such as ratings or buying behavior, and (ii) the attribute information about the users and items such as textual profiles or relevant keywords. Methods that use the former are referred to as *collaborative filtering* methods, whereas methods that use the latter are referred to as *content-based recommender* methods. Note that content-based systems also use the ratings matrices in most cases, although the model is usually focused on the ratings of a single user rather than those of all users. In *knowledge-based* recommender systems, the recommendations are based on explicitly specified *user requirements*. Instead of using historical rating or buying data, external knowledge bases and constraints are used to create the recommendation. Some recommender systems combine these different aspects to create *hybrid* systems. Hybrid systems can combine the strengths of various types of recommender systems to create techniques that can perform more robustly in a wide variety of settings. In the following, we will discuss these basic models briefly, and also provide pointers to the relevant chapters in the book where they are discussed.

1.3.1 Collaborative Filtering Models

Collaborative filtering models use the collaborative power of the ratings provided by multiple users to make recommendations. The main challenge in designing collaborative filtering methods is that the underlying ratings matrices are *sparse*. Consider an example of a movie application in which users specify ratings indicating their like or dislike of specific movies. Most users would have viewed only a small fraction of the large universe of available movies. As a result, most of the ratings are unspecified. The specified ratings are also referred to as *observed ratings*. Throughout this book, the terms “specified” and “observed” will be used in an interchangeable way. The unspecified ratings will be referred to as “unobserved” or “missing.”

The basic idea of collaborative filtering methods is that these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items. For example, consider two users named Alice and Bob, who have very similar tastes. If the ratings, which both have specified, are very similar, then their similarity can be identified by the underlying algorithm. In such cases, it is very likely that the ratings in which only one of them has specified a value, are also likely to be similar. This similarity can be used to make inferences about incompletely specified values. Most of the models for collaborative filtering focus on leveraging either inter-item correlations or inter-user correlations for the prediction process. Some models use both types of correlations. Furthermore, some models use carefully designed optimization techniques to create a training model in much the same way a classifier creates a training model from the labeled data. This model is then used to impute the missing values in the matrix, in the same way that a classifier imputes the missing test labels. There are two types of methods that are commonly used in collaborative filtering, which are referred to as memory-based methods and model-based methods:

1. *Memory-based methods:* Memory-based methods are also referred to as *neighborhood-based collaborative filtering algorithms*. These were among the earliest collaborative filtering algorithms, in which the ratings of user-item combinations are predicted on the basis of their neighborhoods. These neighborhoods can be defined in one of two ways:

   - *User-based collaborative filtering:* In this case, the ratings provided by like-minded users of a *target* user A are used in order to make the recommendations for A. Thus, the basic idea is to determine users, who are similar to the target user A, and recommend ratings for the unobserved ratings of A by computing weighted averages of the ratings of this peer group. Therefore, if Alice and Bob have rated movies in a similar way in the past, then one can use Alice's observed ratings on the movie *Terminator* to predict Bob's unobserved ratings on this movie. In general, the k most similar users to Bob can be used to make rating predictions for Bob. Similarity functions are computed between the *rows* of the ratings matrix to discover similar users.

   - *Item-based collaborative filtering:* In order to make the rating predictions for target item B by user A, the first step is to determine a set S of items that are most similar to target item B. The ratings in item set S, which are specified by A, are used to predict whether the user A will like item B. Therefore, Bob's ratings on similar science fiction movies like *Alien* and *Predator* can be used to predict his rating on *Terminator*. Similarity functions are computed between the *columns* of the ratings matrix to discover similar items.

   The advantages of memory-based techniques are that they are simple to implement and the resulting recommendations are often easy to explain. On the other hand, memory-based algorithms do not work very well with sparse ratings matrices. For example, it might be difficult to find sufficiently similar users to Bob, who have rated *Gladiator*. In such cases, it is difficult to robustly predict Bob's rating of *Gladiator*. In other words, such methods might lack full *coverage* of rating predictions. Nevertheless, the lack of coverage is often not an issue, when only the top-k items are required. Memory-based methods are discussed in detail in Chapter 2.

2. *Model-based methods:* In model-based methods, machine learning and data mining methods are used in the context of predictive models. In cases where the model is parameterized, the parameters of this model are learned within the context of an optimization framework. Some examples of such model-based methods include decision trees, rule-based models, Bayesian methods and latent factor models. Many of these methods, such as latent factor models, have a high level of coverage even for sparse ratings matrices. Model-based collaborative filtering algorithms are discussed in Chapter 3.

Even though memory-based collaborative filtering algorithms are valued for their simplicity, they tend to be heuristic in nature, and they do not work well in all settings. However, the distinction between memory-based and model-based methods is somewhat artificial, because

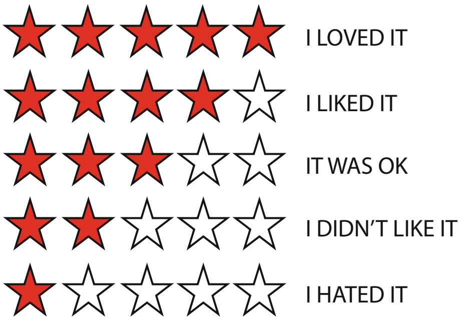
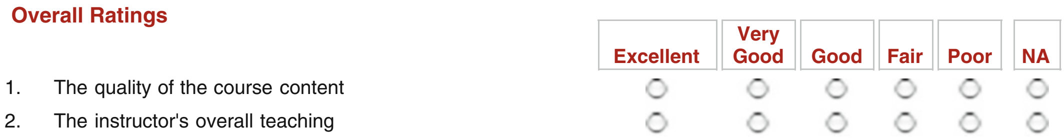


Figure 1.1: Example of 5-point interval ratings



Overall Ratings

		Excellent	Very Good	Good	Fair	Poor	NA
1.	The quality of the course content	○	○	○	○	○	○
2.	The instructor's overall teaching	○	○	○	○	○	○

Figure 1.2: Example of ordinal ratings used in Stanford University course evaluations

memory-based methods can also be considered similarity-based models, albeit heuristic ones. In section 2.6 of Chapter 2, it will also be shown that some variations of neighborhood-based methods can be formally expressed as regression-based models. Latent factor models were popularized in later years as a result of the Netflix Prize contest, although similar algorithmswere proposed much earlier in the context of (generic) incomplete data sets [24]. Recently, it was shown that some combinations of memory-based and model-based methods [309] provide very accurate results.

1.3.1.1 Types of Ratings

The design of recommendation algorithms is influenced by the system used for tracking ratings. The ratings are often specified on a scale that indicates the specific level of like or dislike of the item at hand. It is possible for ratings to be continuous values, such as in the case of the Jester joke recommendation engine [228, 689], in which the ratings can take on any value between -10 and 10. This is, however, relatively rare. Usually, the ratings are interval-based, where a discrete set of ordered numbers are used to quantify like or dislike. Such ratings are referred to as *interval*-based ratings. For example, a 5-point rating scale might be drawn from the set $\{-2, -1, 0, 1, 2\}$, in which a rating of -2 indicates an extreme dislike, and a rating of 2 indicates a strong affinity to the item. Other systems might draw the ratings from the set $\{1, 2, 3, 4, 5\}$.

The number of possible ratings might vary with the system at hand. The use of 5-point, 7-point, and 10-point ratings is particularly common. The 5-star ratings system, illustrated in Figure 1.1, is an example of interval ratings. Along each of the possible ratings, we have indicated a semantic interpretation of the user's level of interest. This interpretation might vary slightly across different merchants, such as Amazon or Netflix. For example, Netflix uses a 5-star ratings system in which the 4-star point corresponds to "*really liked it*," and the central 3-star point corresponds to "*liked it*." Therefore, there are three favorable ratings and two unfavorable ratings in Netflix, which leads to an *unbalanced rating scale*. In some cases, there may be an even number of possible ratings, and the neutral rating might be missing. This approach is referred to as a *forced choice rating system*.

One can also use ordered categorical values such as $\{\text{Strongly Disagree}, \text{Disagree}, \text{Neutral}, \text{Agree}, \text{Strongly Agree}\}$ in order to achieve the same goals. In general, such ratings are referred to as *ordinal* ratings, and the term is derived from the concept of ordinal attributes. An example of ordinal ratings, used in Stanford University course evaluation forms, is illustrated in Figure 1.2. In *binary* ratings, the user may represent only a like or dislike for the item and nothing else. For example, the ratings may be 0, 1, or unspecified values. The unspecified values need to be predicted to 0-1 values. A special case of ratings is that of *unary* ratings, in which there is a mechanism for a user to specify a liking for an item but no mechanism to specify a dislike. Unary ratings are particularly common, especially in the case of *implicit feedback data sets* [259, 260, 457]. In these cases, customer preferences are derived from their activities rather than their explicitly specified ratings. For example, the buying behavior of a customer can be converted to unary ratings. When a customer buys an item, it can be viewed as a preference for the item. However, the act of not buying an item from a large universe of possibilities does not always indicate a dislike. Similarly, many social networks, such as Facebook, use "*like*" buttons, which provide the ability to express liking for an item. However, there is no mechanism to specify dislike for an item. The implicit feedback setting can be viewed as the matrix completion analog of the positive-unlabeled (PU) learning problem in data classification [259].

Examples of Explicit and Implicit Ratings

A quantitative example of explicit ratings is illustrated in Figure 1.3(a). In this case, there are 6 users, labeled $U_1 \ldots U_6$, and 6 movies with specified titles. Higher ratings indicate more positive feedback in Figure 1.3(a). The missing entries correspond to unspecified preferences. The example of this figure represents a small toy example. In general, the ratings could be represented as an $m \times n$ matrix, where m and n are typically very large and may range in the order of hundreds of thousands. Even though this particular example uses a 6×6 matrix, the values of m and n are typically not the same in real-world scenarios. A ratings matrix is sometimes referred to as a *utility matrix*, although the two may not always be the same. Strictly speaking, when the utility refers to the amount of profit, then the utility of a user-item combination refers to the amount of profit incurred by recommending that item to the particular user. While utility matrices are often set to be the same as the ratings matrices, it is possible for the application to explicitly transform the ratings to utility values based on domain-specific criteria. All collaborative filtering algorithms are then applied to the utility matrix rather than the ratings matrix. However, such an approach is rarely used in practice, and most collaborative filtering algorithms work directly with the ratings matrix.

An example of a unary ratings matrix is illustrated in Figure 1.3(b). For cases in which the ratings are unary, the matrix is referred to as a *positive preference utility matrix* because it allows only the specification of positive preferences. The two matrices in Figure 1.3 have the same set of observed entries, but they provide very different insights. For example, the users U_1 and U_3 are very different in Figure 1.3(a) because they have very different ratings for their mutually specified entries. On the other hand, these users would be considered very similar in Figure 1.3(b) because these users have expressed a positive preference for the same items. The ratings-based utility provides a way for users to express negative preferences for items. For example, user U_1 does not like the movie *Gladiator* in Figure 1.3(a). There is no mechanism to specify this in the positive-preference utility matrix of Figure 1.3(b) beyond

	GLADITOR	GODFTHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

(a) Ordered ratings

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

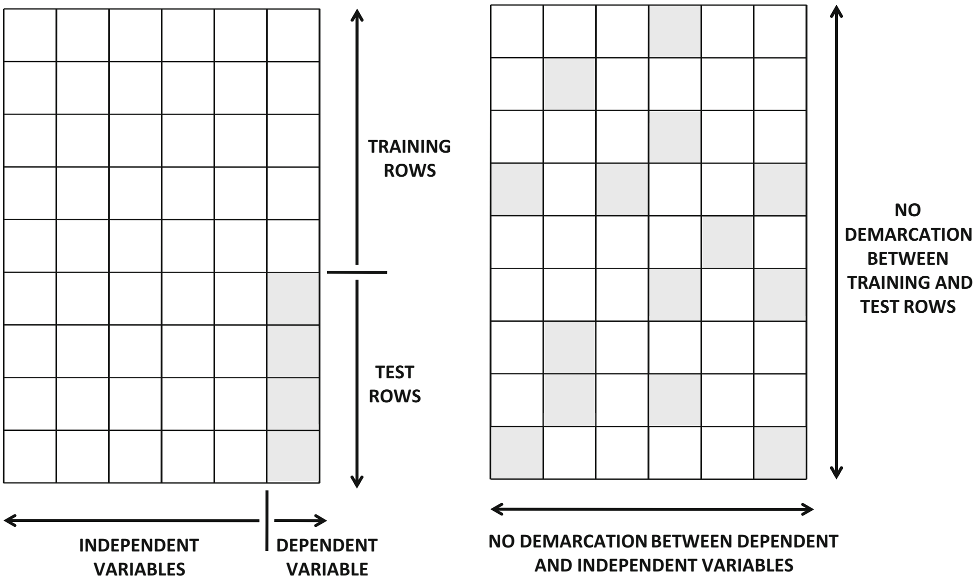
(b) Unary ratings

Figure 1.3: Examples of utility matrices

a relatively ambiguous missing entry. In other words, the matrix in Figure 1.3(b) is less expressive. While Figure 1.3(b) provides an example of a binary matrix, it is possible for the nonzero entries to be arbitrary positive values. For example, they could correspond to the quantities of items bought by the different users. In general, unary matrices are created by user *actions* such as buying an item, and are therefore also referred to as implicit feedback matrices.

Unary ratings have a significant effect on the recommendation algorithm at hand, because no information is available about whether a user dislikes an item. In the case of unary matrices, it is often recommended [260] to perform the analysis in a simple way by treating the missing entries as 0s in the initial phase. However, the final predicted value by the learning algorithm might be much larger than 0, especially if the item matches user interests. The recommended items are therefore based on the entries with the largest positive prediction error over the initial "zero" assumption. In fact, if the missing entries are not substituted with 0s, significant overfitting is possible. This type of overfitting is an artifact of the fact that there is often not a sufficient level of discrimination between the various observed values of the ratings. In explicit feedback matrices, ratings correspond to (highly discriminated) *preferences*, whereas in implicit feedback matrices, ratings correspond to (less discriminated) *confidences*. In a later chapter, we will provide a specific example of overfitting with implicit feedback matrices when missing entries are not treated as zeros (cf. section 3.6.6.2 of Chapter 3).

Pre-substitution of missing ratings is not recommended in explicit ratings matrices. In explicit ratings matrices with both likes and dislikes, the substitution of missing entries with any value (such as 0 or the row/column/data mean) always leads to a significant amount of bias in the analysis. In the unary case, substituting missing entries with 0s also leads to some bias [457, 467, 468], although it is often small because the default assumption in implicit feedback data, such as buying data, is that the user will not buy most of the items. One is often willing to live with this bias in the unary case, because a significant amount of overfitting is reduced by the substitution. There are also some interesting computational effects of such choices. These trade-offs are discussed in Chapters 2 and 3.


TRAINING ROWS
TEST ROWS
INDEPENDENT VARIABLES
DEPENDENT VARIABLE
NO DEMARCATION BETWEEN TRAINING AND TEST ROWS
NO DEMARCATION BETWEEN DEPENDENT AND INDEPENDENT VARIABLES

(a) Classification

(b) Collaborative filtering

Figure 1.4: Comparing the traditional classification problem with collaborative filtering. Shaded entries are missing and need to be predicted.

1.3.1.2 Relationship with Missing Value Analysis

Collaborative filtering models are closely related to missing value analysis. The traditional literature on missing value analysis studies the problem of imputation of entries in an incompletely specified data matrix. Collaborative filtering can be viewed as a (difficult)special case of this problem in which the underlying data matrix is very *large* and *sparse*. A detailed discussion of methods for missing value analysis in the statistical literature may be found in [362]. Many of these methods can also be used for recommender systems, although some of them might require specialized adaptations for very large and sparse matrices. In fact, some of the recent classes of models for recommender systems, such as latent factor models, were studied earlier in the context of missing value analysis [24]. Similar methods were independently proposed in the context of recommender systems [252, 309, 313, 500, 517, 525]. In general, many classical missing value estimation methods [362] can also be used for collaborative filtering.

1.3.1.3 Collaborative Filtering as a Generalization of Classification and Regression Modeling

Collaborative filtering methods can be viewed as generalizations of classification and regression modeling. In the classification and regression modeling problems, the class/dependent variable can be viewed as an attribute with missing values. Other columns are treated as features/independent variables. The collaborative filtering problem can be viewed as a generalization of this framework because any column is allowed to have missing values rather than (only) the class variable. In the recommendation problem, a clear distinction does not exist between class variables and feature variables because each feature plays the dual role of a dependent and independent variable. This distinction exists in the classification problem *only* because the missing entries are restricted to a special column. Furthermore, there is no distinction between training and test rows in collaborative filtering because any row might contain missing entries. Therefore, it is more meaningful to speak of training and test *entries* in collaborative filtering rather than training and test *rows*. Collaborative filtering is a generalization of classification/regression modeling in which the prediction is performed in entry-wise fashion rather than row-wise fashion. This relationship between classification/regression modeling and collaborative filtering is important to keep in mind because many principles of classification and regression modeling methods can be generalized to recommender systems. The relationship between the two problems is illustrated in Figure 1.4. This figure is particularly useful in relating collaborative filtering with classification, and it will be revisited multiple times in this book. wherever the similarities between these two problems are leveraged in some way for algorithmic or theoretical development.

The matrix completion problem also shares a number of characteristics with the *transductive* setting in classification and regression. In the transductive setting, the test instances are also included in the training process (typically with the use of a semisupervised algorithm), and it is often hard to make predictions for test instances that are not available at the time of training. On the other hand, models in which predictions can be easily made for new instances are referred to as *inductive*. For example, a naive Bayes model in classification is inherently inductive because one can easily use it to predict the label of a test instance for which the features were not known at the time of building the Bayes model.

The setting for matrix completion is inherently transductive because the training and test data are tightly integrated with one another in the $m \times n$ ratings matrix R, and many models cannot easily predict ratings for out-of-sample users and/or items. For example, if John is added to the ratings matrix (with many specified ratings) after the collaborative filtering model has already been constructed, many off-the-shelf methods will not be able to make predictions for John. This is especially true for model-based collaborative filtering methods. However, some recent matrix completion models have also been designed to be inductive in which ratings can be predicted for out-of-sample users and/or items.

1.3.2 Content-Based Recommender Systems

In content-based recommender systems, the descriptive attributes of items are used to make recommendations. The term "content" refers to these descriptions. In content-based methods, the ratings and buying behavior of users are combined with the content information available in the items. For example, consider a situation where John has rated the movie *Terminator* highly, but we do not have access to the ratings of other users. Therefore, collaborative filtering methods are ruled out. However, the item description of *Terminator* contains similar genre keywords as other science fiction movies, such as *Alien* and *Predator*. In such cases, these movies can be recommended to John.

In content-based methods, the item descriptions, which are labeled with ratings, are used as training data to create a user-specific classification or regression modeling problem. For each user, the training documents correspond to the descriptions of the items she has bought or rated. The class (or dependent) variable corresponds to the specified ratings or buying behavior. These training documents are used to create a classification or regression model, which is *specific* to the user at hand (or *active* user). This user-specific model is used to predict whether the corresponding individual will like an item for which her rating or buying behavior is unknown.

Content-based methods have some advantages in making recommendations for new items, when sufficient rating data are not available for that item. This is because other items with similar attributes might have been rated by the active user. Therefore, the supervised model will be able to leverage these ratings in conjunction with the item attributes to make recommendations even when there is no history of ratings for that item.

Content-based methods do have several disadvantages as well:

1. In many cases, content-based methods provide *obvious* recommendations because of the use of keywords or content. For example, if a user has never consumed an item with a particular set of keywords, such an item has no chance of being recommended. This is because the constructed model is specific to the user at hand, and the community knowledge from similar users is not leveraged. This phenomenon tends to reduce the diversity of the recommended items, which is undesirable.

2. Even though content-based methods are effective at providing recommendations for new *items*, they are not effective at providing recommendations for new *users*. This is because the training model for the target user needs to use the history of her ratings. In fact, it is usually important to have a large number of ratings available for the target user in order to make robust predictions without overfitting.

Therefore, content-based methods have different trade-offs from collaborative filtering systems.

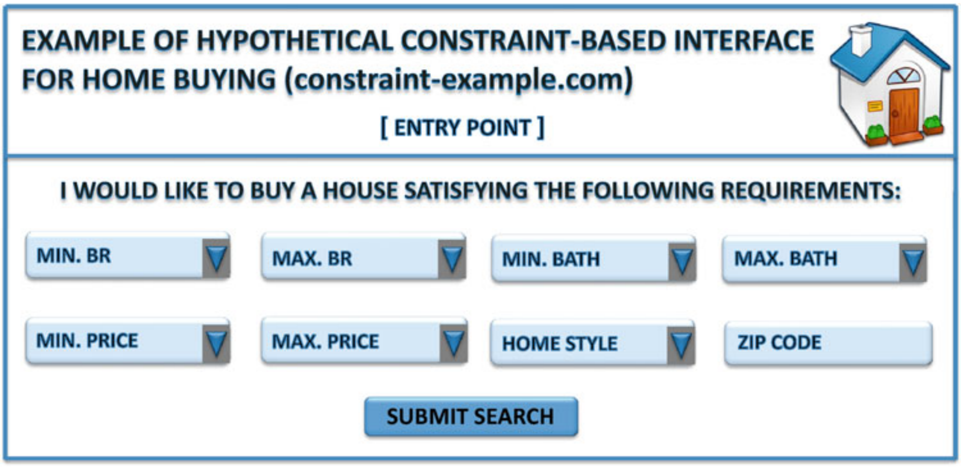
Although the aforementioned description provides the conventional learning-based view of content-based methods, a broader view of these methods is sometimes used. For example, users can specify relevant keywords in their own profiles. These profiles can be matched with item descriptions in order to make recommendations. Such an approach does not use ratings in the recommendation process, and it is therefore useful in cold-start scenarios. However, such methods are often viewed as a distinct class of recommender systems, known as *knowledge-based systems*, because the similarity metrics are often based on domain knowledge. Knowledge-based recommender systems are often considered to be closely related to content-based recommender systems, and it is sometimes questioned whether a clear demarcation exists between the two classes of methods [558]. Methods for content-based recommender systems are discussed in Chapter 4.

1.3.3 Knowledge-Based Recommender Systems

Knowledge-based recommender systems are particularly useful in the context of items that are not purchased very often. Examples include items such as real estate, automobiles, tourism requests, financial services, or expensive luxury goods. In such cases, sufficient ratings may not be available for the recommendation process. As the items are bought rarely, and with different types of detailed options, it is difficult to obtain a sufficient number of ratings for a specific instantiation (i.e., combination of options) of the item at hand. This problem is also encountered in the context of the cold-start problem, when sufficient ratings are not available for the recommendation process. Furthermore, the nature of consumer preferences may evolve over time when dealing with such items. For example, the model of a car may evolve significantly over a few years, as a result of which the preferences may show a corresponding evolution. In other cases, it might be difficult to fully capture user interest with historical data such as ratings. A particular item may have attributes associated with it that correspond to its various properties, and a user may be interested only in items with specific properties. For example, cars may have several makes, models,

Table 1.2: The conceptual goals of various recommender systems

Approach	Conceptual Goal	Input
Collaborative	Give me recommendations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings + community ratings
Content-based	Give me recommendations based on the content (attributes) I have favored in my past ratings and actions.	User ratings + item attributes
Knowledge-based	Give me recommendations based on my explicit specification of the kind of content (attributes) I want.	User specification + item attributes + domain knowledge



EXAMPLE OF HYPOTHETICAL CONSTRAINT-BASED INTERFACE FOR HOME BUYING (constraint-example.com)

[ENTRY POINT]

I WOULD LIKE TO BUY A HOUSE SATISFYING THE FOLLOWING REQUIREMENTS:

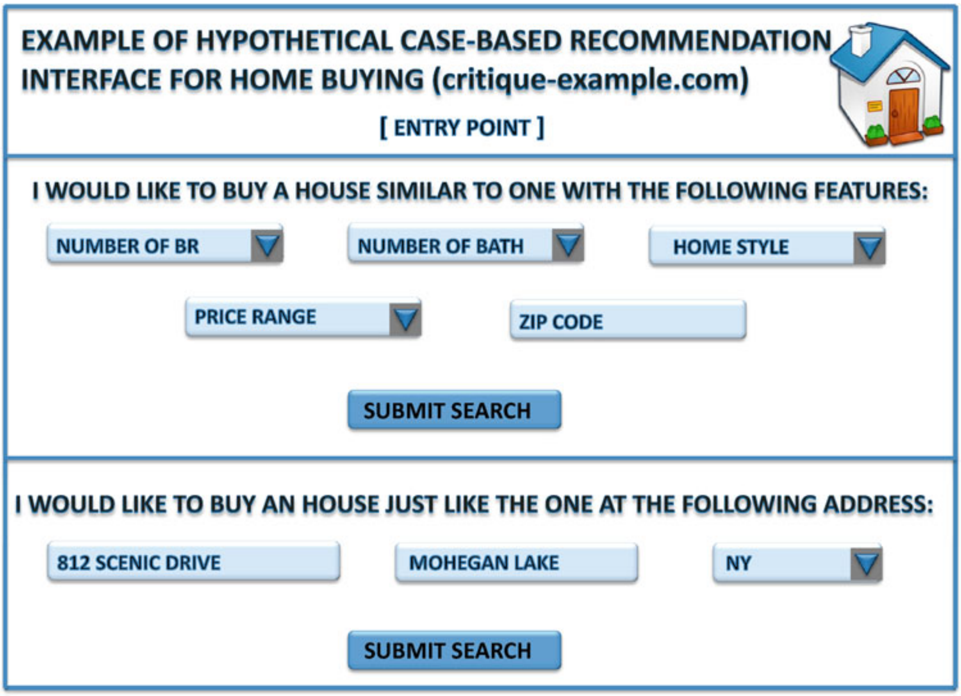
Figure 1.5: A hypothetical example of an initial user interface for a constraint-based recommender)

colors, engine options, and interior options, and user interests may be regulated by a very specific combination of these options. Thus, in these cases, the item domain tends to be *complex* in terms of its varied properties, and it is hard to associate sufficient ratings with the large number of combinations at hand.

Such cases can be addressed with knowledge-based recommender systems, in which ratings are not used for the purpose of recommendations. Rather, the recommendation processis performed on the basis of similarities between customer requirements and item descriptions, or the use of constraints specifying user requirements. The process is facilitated with the use of *knowledge bases*, which contain data about rules and similarity functions to use during the retrieval process. In fact, the knowledge bases are so important to the effective functioning of these methods that the approach takes its name from this fact. The explicit specification of requirements results in greater control of users over the recommendation process. In both collaborative and content-based systems, recommendations are decided entirely by either the user's past actions/ratings, the action/ratings of her peers, or a combination of the two. Knowledge-based systems are unique in that they allow the users to *explicitly specify what they want.* This difference is illustrated in Table 1.2.

Knowledge-based recommender systems can be classified on the basis of the type of the interface (and corresponding knowledge) used to achieve the aforementioned goals:

1. *Constraint-based recommender systems:* In constraint-based systems [196, 197], users typically specify requirements or constraints (e.g., lower or upper limits) on the item



EXAMPLE OF HYPOTHETICAL CASE-BASED RECOMMENDATION INTERFACE FOR HOME BUYING (critique-example.com)

[ENTRY POINT]

I WOULD LIKE TO BUY A HOUSE SIMILAR TO ONE WITH THE FOLLOWING FEATURES:

I WOULD LIKE TO BUY AN HOUSE JUST LIKE THE ONE AT THE FOLLOWING ADDRESS:

812 SCENIC DRIVE

MOHEGAN LAKE

NY

Figure 1.6: A hypothetical example of an initial user interface for a case-based recommender)

attributes. An example of such an interface is illustrated in Figure 1.5. Domain-specific rules are used to match the user requirements to item attributes. These rules represent the domain-specific knowledge used by the system. Such rules could take the form of domain-specific constraints on the item attributes (e.g., "*Cars before year 1970 do not have cruise control.*"). Furthermore, constraint-based systems often create rules relating user attributes to item attributes (e.g., "*Older investors do not invest in ultra high-risk products.*"). In such cases, user attributes may also be specified in the search process. Depending on the number and type of returned results, the user might have an opportunity to modify their original requirements. For example, they might relax some of their constraints when too few results are returned, or they might add more constraints. This search process is interactively repeated until the user arrives at her desired results.

2. *Case-based recommender systems:* In case-based recommender systems [102, 116, 377, 558], specific cases are specified by the user as targets or anchor points. Similarity metrics are defined on the item attributes to retrieve similar items to these cases. An example of such an interface is illustrated in Figure 1.6. The similarity metrics are often carefully defined in a domain-specific way. Therefore, the similarity metrics form the domain knowledge that is used in such systems. The returned results are often used as new target cases with some interactive modifications by the user. For example, when a user sees a returned result, which is almost similar to what they want, they might re-issue a query with that target, but with some of the attributes changed to the user's liking. This interactive process is used to guide the user towards items of interest.

Note that in both cases, the system provides an opportunity to the user to change their specified requirements. However, the way in which this is done is different in the two cases. In case-based systems, examples (or *cases*) are used as anchor points to guide the search in combination with similarity metrics. Critiquing interfaces are particularly popular for expressing feedback in such systems, where users iteratively modify one or more attributes of a preferred item in each iteration. In constraint-based systems, rules (or *constraints*) are used to guide the search. The form of the guidance may often take the form of search-based systems, where users specify their constraints with a search-based interface.

How is the interactivity in knowledge-based recommender systems achieved? This guidance takes place through one or more of the following methods:

1. *Conversational systems:* In this case, the user preferences are determined iteratively in the context of a feedback loop. The main reason for this is that the item domain is complex and the user preferences can be determined only in the context of an iterative conversational system.

2. *Search-based systems:* In search-based systems, user preferences are elicited by using a preset sequence of questions such as the following: "Do you prefer a house in a suburban area or within the city?" In some cases, specific search interfaces may be set up in order to provide the ability to specify user constraints.

3. *Navigation-based recommendation:* In navigation-based recommendation, the user specifies a number of change requests to the item being currently recommended. Through an iterative set of change requests, it is possible to arrive at a desirable item. An example of a change request specified by the user, when a specific house is being recommended is as follows: "I would like a similar house about 5 miles west of the currently recommended house." Such recommender systems are also referred to as *critiquing recommender systems* [417].

It is noteworthy that both knowledge-based and content-based systems depend significantly on the attributes of the items. Because of their use of content-attributes, knowledge-based systems inherit some of the same disadvantages as content-based systems. For example, just like content-based systems, the recommendations in knowledge-based systems can sometimes be obvious because the use of community (i.e., peer) ratings is not leveraged. In fact, knowledge-based systems are sometimes considered to be the "cousins" of content-based systems [558]. The main difference is that content-based systems learn from *past user behavior*, whereas knowledge-based recommendation systems recommend based on active user *specification of their needs and interests*. Therefore, in most of the recommendation literature, knowledge-based recommenders are considered to be a distinct category from content-based recommenders. These distinctions are based both on the goals of such systems and the kind of input data used (see Table 1.2). The different forms of knowledge-based recommender systems are discussed in Chapter 5.

1.3.3.1 Utility-Based Recommender Systems

In utility-based recommender systems, a utility function is defined on the product features in order to compute the probability of a user liking the item [239]. The central challenge in utility-based methods is in defining an appropriate utility function for the user at hand. It is noteworthy that all recommender schemes, whether collaborative, content-based, or knowledge-based methods, implicitly rank the recommended items on the basis of their perceived value (or *utility*) for the target user. In utility-based systems, this utility value is based on a function that is known *a priori*. In this sense, such functions can be viewed as a kind of external knowledge. Therefore, utility-based systems can be viewed as a specific case of knowledge-based recommender systems. In fact, it will be shown in Chapter 5 that utility functions are used frequently in various ways for ranking items in knowledge-based recommender systems.

1.3.4 Demographic Recommender Systems

In demographic recommender systems, the demographic information about the user is leveraged to learn classifiers that can map specific demographics to ratings or buying propensities. An early recommender system, referred to as *Grundy* [508], recommended books based on the library of manually assembled stereotypes. The characteristics of the user were collected with the use of an interactive dialogue. The work in [320] observed that the demographic groups from marketing research can be used to recommend items. Another work [475] makes Web page recommendations on the basis of the demographic characteristics of users that have rated a particular page highly. In many cases, demographic information can be combined with additional *context* to guide the recommendation process. This approach is related to the methodology of *context-sensitive recommender systems*. Some of these methods are discussed in section 8.5.3 of Chapter 8.

More recent techniques have focused on using classifiers for making recommendations. One of the interesting systems in this respect was a technique that extracted features from user home pages in order to predict their likelihood of liking certain restaurants. Rule-based classifiers [31, 32] are often used to relate the demographic profile to buying behavior in an interactive way. While the approach in [31, 32] was not specifically used to recommend specific items, it can easily be paired with a recommender system. Such recommender systems are not very different from the vanilla classification and regression modeling problem, in which feature variables correspond to the demographic profiles and the dependent variables correspond to the ratings or to the buying behavior. Although demographic recommender systems do not usually provide the best results on a stand-alone basis, they add significantly to the power of other recommender systems as a component of hybrid or ensemble models. Demographic techniques are sometimes combined with knowledge-based recommender systems to increase their robustness.

1.3.5 Hybrid and Ensemble-Based Recommender Systems

The three aforementioned systems exploit different sources of input, and they may work well in different scenarios. For example, collaborative filtering systems rely on community ratings, content-based methods rely on textual descriptions and the target user's own ratings, and knowledge-based systems rely on interactions with the user in the context of knowledge bases. Similarly, demographic systems use the demographic profiles of the users to make recommendations. It is noteworthy that these different systems use different types of input, and have different strengths and weaknesses. Some recommender systems, such as knowledge-based systems, are more effective in cold-start settings where a significant amount of data is not available. Other recommender systems, such as collaborative methods, are more effective when a lot of data is available.

In many cases where a wider variety of inputs is available, one has the flexibility of using different types of recommender systems for the same task. In such cases, many opportunities exist for hybridization, where the various aspects from different types of systems are combined to achieve the best of all worlds. Hybrid recommender systems are closely related to the field of ensemble analysis, in which the power of multiple types of machine learning algorithms is combined to create a more robust model. Ensemble-based recommender systems are able to combine not only the power of multiple data sources, but they are also able to improve the effectiveness of a particular class of recommender systems (e.g., collaborative systems) by combining multiple models of the same type. This scenario is not very different from that of ensemble analysis in the field of data classification. Chapter 6 studies various hybridization strategies for recommender systems.

1.3.6 Evaluation of Recommender Systems

Given a set of recommendation algorithms, how well do they perform? How can we evaluate their relative effectiveness? Recommender systems share several conceptual similarities with the classification and regression modeling problem. In classification and regression modeling, the missing class variable needs to be predicted from the feature variables. In recommender systems, any of the matrix entries may be missing and need to be predicted in a data-driven way from the observed entries in the remaining matrix. In this sense, the recommendation problem can be viewed as a generalization of the classification problem. Therefore, many of the models used for evaluation of classifiers can be used for evaluating recommender systems, albeit with some modifications. There are significant variations in the evaluation techniques used for different aspects of recommender systems, such as rating prediction or ranking. The former is closely related to classification and regression modeling, whereas the latter is closely related to the evaluation of retrieval effectiveness in search and information retrieval applications. Evaluation methods for recommender systems are discussed in detail in Chapter 7.

1.4 Domain-Specific Challenges in Recommender Systems

In different domains, such as temporal data, location-based data, and social data, the context of the recommendation plays a critical role. Therefore, the notion of *contextual recommender systems* was developed to address the additional side information that arises in these domains. This notion is used with different modifications for various types of data, such as temporal data, location data, or social data.

1.4.1 Context-Based Recommender Systems

Context-based or context-*aware* recommender systems take various types of contextual information into account, while making recommendations. Such contextual information could include time, location, or social data. For example, the types of clothes recommended by a retailer might depend both on the season and the location of the customer. Another example is the case in which a particular type of festival or holiday affects the underlying customer activity.

It has generally been observed that the use of such contextual information can greatly improve the effectiveness of the recommendation process. Context-based recommender systems are incredibly powerful because the underlying ideas are relevant to a wide variety of domain-specific settings. In fact, a recurring theme throughout the later chapters of the book, will be the use of a *multidimensional model* [7] for context-specific recommendations in different domain-specific settings. Context-aware recommender systems will be discussed in Chapter 8 in a general sense. However, individual aspects of the context, such as time, location, and social information, are studied in detail in other chapters. A general review of these different aspects is provided below.

1.4.2 Time-Sensitive Recommender Systems

In many settings, the recommendations for an item might evolve with time. For example, the recommendations for a movie may be very different at the time of release from the recommendations received several years later. In such cases, it is extremely important to incorporate temporal knowledge in the recommendation process. The temporal aspect in such recommender systems can be reflected in several ways:

1. The rating of an item might evolve with time, as community attitudes evolve and the interests of users change over time. User interests, likes, dislikes, and fashions inevitably evolve with time.

2. The rating of an item might be dependent on the specific time of day, day of week, month, or season. For example, it makes little sense to recommend winter clothing during the summer, or raincoats during the dry season.

The first type of recommender system is created by incorporating time as an explicit parameter in collaborative filtering systems. The second type can be viewed as a special case of context-based recommender systems. Temporal recommender systems are challenging because of the fact that the matrix of ratings is sparse, and the use of specific temporal context aggravates the sparsity problem. Therefore, it is particularly important to have access to large data sets in these settings.

Another common setting is that of implicit feedback data sets such as Web click-streams. The user activity on the Web and other internet platforms creates a lot of useful data that can be mined to make recommendations about future activity. In such cases, discrete sequential pattern mining and Markov models are helpful. The problem of time-sensitive recommendation is discussed in detail in Chapter 9.

1.4.3 Location-Based Recommender Systems

With the increasing popularity of GPS-enabled mobile phones, consumers are often interested in location-based recommendations. For example, a traveling user may wish to determine the closest restaurant based on her previous history of ratings for other restaurants. In general, the recommendation of *places* always has a location aspect built into it. An example of such a system is Foursquare[2], which recommends various types of places such as restaurants or nightlife venues. There are two types of spatial locality that are common to such systems:

1. *User-specific locality:* The geographical location of a user has an important role in her preferences. For example, a user from Wisconsin might not have the same movie preferences as a user from New York. This type of locality is referred to as *preference locality.*

[2] http://foursquare.com

2. *Item-specific locality:* The geographical location of an item (e.g., restaurant) might have an impact on the relevance of the item, depending on the current location of the user. Users are generally not willing to travel very far from their current location. This type of locality is referred to as *travel locality.*

The algorithms for preference locality and travel locality are quite different. The former are closer to context-sensitive systems, whereas the latter are usually designed as ad hoc heuristics. Location-based recommender systems have witnessed an increasing interest in recent years because of the increasing prevalence of mobile phones and other GPS-enabled devices. Location-based recommender systems are discussed in detail in Chapter 9.

1.4.4 Social Recommender Systems

Social recommender systems are based on network structures, social cues and tags, or a combination of these various network aspects. In general, the recommender systems that are based on social cues and tags are slightly different from those that are based purely on structural aspects. Recommender systems, which are based purely on structural aspects, are used to suggest nodes and links within the network itself. On the other hand, social recommender systems may be also be used to recommend various products with the use of social cues. Both these forms of recommender systems will be studied in this book. However, these forms of recommendation are sufficiently different that they will be studied in different chapters of this book. It is important to note that the utility of structural recommender systems extends beyond social networks, because such methods are applied to various types of Web-enabled networks.

1.4.4.1 Structural Recommendation of Nodes and Links

Various types of networks, including social networks, are composed of nodes and links. In many cases, it is desirable to recommend nodes and links. For example, a personalized Web search may require a recommendation of material which is related to a particular topic. Since the Web can be viewed as a graph, such methods can be viewed as a node recommendation problem. The problem of node recommendation is closely related to the problem of Web search. In fact, both problems require the use of various forms of ranking algorithms. A key component of these methods is the use of the *PageRank* algorithm, although the personalization of such algorithms is more closely related to recommendation algorithms. Therefore, such algorithms are also referred to as personalized *PageRank* algorithms. In cases where examples of nodes of interest are available, such nodes can be used as training data in order to determine other nodes of interest. This problem is referred to as *collective classification.* A closely related problem is that of the link recommendation or link prediction problem, where it is desirable to suggest friends (or potential links) for a user in a social network. The link prediction problem also has numerous applications beyond social networks. Interestingly, the problems of ranking, collective classification, and link recommendation are closely related. In fact, solutions to one problem are often used as subroutines for other problems. For example, ranking and link prediction methods are often used for traditional product recommendations in user-item graphs. In fact, these methods can be used to perform recommendations in many problem settings, which can be transformed into graphs. Methods for node and link recommendations are discussed in Chapter 10.

1.4.4.2 Product and Content Recommendations with Social Influence

Many forms of product and content recommendation are performed with the help of network connections and other social cues. This problem is also referred to as *viral marketing.* In viral marketing, products are recommended with the use of word-of-mouth systems. In order to achieve this goal, it is important to be able to determine influential and topically relevant entities in the network. This problem is referred to as *influence analysis* in social networks [297]. Many variations of this problem have been proposed, in which the influencers are found in a topically sensitive way, in the social stream scenario. For example, determining the influential users in a Twitter stream for specific topics may be very useful for viral marketing. In other cases, social cues are harvested from social networks in order to make recommendations. These methods for discussed in Chapter 10.

1.4.4.3 Trustworthy Recommender Systems

Many social media sites, such as Epinions [705] or Slashdot [706], allow users to express their trust and distrust in one another, either in a direct way, or through various feedback mechanisms. For example, users can express their trust or distrust in reviews of other users, or they may directly specify their trust or distrust relationships with other users. This trust information is very useful for making more robust recommendations. For example, it is evident that a user-based neighborhood method should be computed with the use of trustworthy peers to obtain robust recommendations. Recent research has shown [221, 588, 616] that the incorporation of trust information can lead to more robust recommendations. Trustworthy recommender systems are presented in Chapter 11.

1.4.4.4 Leveraging Social Tagging Feedback for Recommendations

Users have numerous methods for incorporating their feedback in recommender systems. The most common form of feedback is *social tagging.* Such forms of feedback are particularly common on content sharing sites on the Web, such as Flickr (photo sharing) [692], last.fm [692] (music sharing), and Bibsonomy [708] (scientific literature sharing). *Tags* are meta-data that users utilize to add short informative keywords to the content. For example, a user on a music site might tag Michael Jackson's *Thriller* album as "*rock.*" Such tags provide useful information about the interests of both the user and the content of the item because the tag is associated with both. The tags serve as useful context for performing the recommendations. Methods for context-sensitive recommendations can be directly used to incorporate this feedback into the recommendation process. Other specialized methods have also been developed for using social tagging feedback in the recommendation process. These methods are discussed in detail in Chapter 11.

1.5 Advanced Topics and Applications

This book will also introduce a number of advanced topics and applications. Most of the these topics are discussed in Chapters 12 and 13, although some of the topics are spread out over the book, where it is appropriate. In this section, we provide a brief introduction to these topics.

1.5.1 The Cold-Start Problem in Recommender Systems

One of the major problems in recommender systems is that the number of initially available ratings is relatively small. In such cases, it becomes more difficult to apply traditional collaborative filtering models. While content-based and knowledge-based methods are more robust than collaborative models in the presence of cold starts, such content or knowledge might not always be available. Therefore, a number of specific methods have been designed to ameliorate the problem of cold start in the context of recommender systems. The susceptibility of various models to the cold-start problem is also highlighted throughout this book, along with possible solutions.

1.5.2 Attack-Resistant Recommender Systems

The use of recommender systems has a significant impact on the sale of various products and services. As a result, the sellers of products and services have significant economic incentives to manipulate the output of recommender systems. One example of such a manipulation would be to submit inflated ratings of their own products to the recommender systems. A malicious rival might submit biased and negative reviews about the products of a competitor. Over the years, numerous sophisticated strategies have been developed for attacking recommender systems. Such attacks are highly undesirable because they reduce the overall effectiveness of the recommender system and reduce the quality of experience for legitimate users. Therefore, methods are needed that enable robust recommendations in the presence of such attacks. Attack methods, including the susceptibility of various types of algorithms to attacks, are discussed in detail in Chapter 12. In addition, Chapter 12 will provide a number of strategies for constructing robust recommender systems in the presence of such attacks.

1.5.3 Group Recommender Systems

An interesting extension of traditional recommender systems is the notion of *group recommender systems* [168]. In such cases, the recommendation system is tailored to recommend a particular activity to a group of users rather than a single user. Examples might include the watching of movie or television by a group [408, 653], the selection of music in a fitness center, or the travel recommendations to a group of tourists. The earliest systems, such as *PolyLens* [168], designed models that aggregated the preferences of individual users in order to create group recommendations. However, the consensus over the years has evolved into designing recommender systems, which are better than the sum of their parts and can take the interactions between the various users into account for designing recommendations [272, 413]. Simple averaging strategies do not work well when groups are heterogeneous and contain users with diverse tastes [653]. This is because users often have an impact on each other's tastes based on phenomena from social psychology, such as *emotional contagion* and *conformity.* Detailed surveys on the subject may be found in [45, 271, 407]. Group recommender systems are discussed in section 13.4 of Chapter 13.

1.5.4 Multi-Criteria Recommender Systems

In multi-criteria systems, ratings might be specified on the basis of different criteria by a single user. For example, a user might rate movies based on the plot, music, special effects, and so on. Such techniques often provide recommendations by modeling the user's utility for an item as a vector of ratings corresponding to various criteria. In multi-criteria recommender systems, one can often obtain misleading results by using only the overall rating in conjunction with a traditional recommender system. For example, if two users have the same overall rating for a movie, but their component ratings for the plot and music are very different, then the two users should not be considered similar from the perspective of a similarity-based collaborative filtering algorithm. In some of the multi-criteria systems, users may not specify an overall rating at all. In such cases, the problem is even more challenging because it is needed to present ranked lists of items to various users on the basis of multiple criteria. Excellent overviews of multi-criteria recommender systems may be found in [11, 398, 604] from various perspectives.

It has been shown [271, 410], that some of the methods for group recommender systems can also be adapted to multi-criteria recommender systems. However, the two topics are generally considered different because they emphasize different aspects of the recommendation process. Methods for multi-criteria recommender systems are discussed in section 13.5 of Chapter 13.

1.5.5 Active Learning in Recommender Systems

A major challenge in recommender systems is the acquisition of sufficient ratings in order to make robust predictions. The sparsity of the ratings matrix continues to be a significant impediment in effective functioning of recommender systems. The acquisition of sufficient ratings can reduce the sparsity problem. A variety of real-world recommender systems have mechanisms to encourage users to enter ratings in order to populate the system. For example, users might be provided incentives to rate certain items. In general, it is often difficult to obtain too many ratings from the single user because of the high cost of the acquisition process. Therefore, one must judiciously select the items to be rated by specific users. For example, if a user has already rated a lot of action movies, then asking the user to rate another action movie does not help much in predicting ratings of other action movies, and it helps even less in predicting ratings of movies belonging to unrelated genres. On the other hand, asking the user to rate movies belonging to less populated genres will help significantly in predicting ratings of movies belonging to that genre. Of course, if a user is asked to rate an unrelated movie, it is not necessary that she will be able to provide feedback because she might not have watched that movie at all. Therefore, there are many interesting trade-offs in the problem of active learning of recommender systems, that are not encountered in other problem domains like classification. A review of active learning methods for recommender systems may be found in [513]. Active learning methods are discussed in section 13.6 of Chapter 13.

1.5.6 Privacy in Recommender Systems

Recommender systems are based heavily on feedback from the users, which might be implicit or explicit. This feedback contains significant information about the interests of the user, and it might reveal information about their political opinions, sexual orientations, and personal preferences. In many cases, such information can be highly sensitive, which leads to privacy concerns. Such privacy concerns are significant in that they impede the release of data necessary for the advancement of recommendation algorithms. The availability of real data is crucial for algorithmic advances. For example, the contribution of the Netflix Prize data set to the recommender systems community is invaluable, in that it can be credited with motivating the development of many state-of-the-art algorithms [373]. In recent years, the topic of privacy has been explored in the context of a wide variety of data mining problems [20]. The recommendation domain is no exception, and numerous privacy-preserving algorithms have been developed [133, 484, 485]. The topic of privacy in recommender systems is discussed in detail in section 13.7 of Chapter 13.

1.5.7 Application Domains

Recommender systems are used in numerous application domains, such as retail, music, content, Web search, querying, and computational advertisements. Some of these domains require specialized methods for adapting recommender systems. In particular, Chapter 13 will study three specific domains corresponding to news recommendations, computational advertising, and reciprocal recommender systems. All these application domains are Web-centric in nature. An important aspect of recommender systems is that they assume the existence of strong user-identification mechanisms in order to track and identify long-term user interests. In many Web domains, mechanisms for strong user identification may not be available. In such cases, direct user of recommendation technology may not be feasible. Furthermore, since new items (advertisements) continually enter and leave the system, certain types of methods such as multi-armed bandits are particularly suitable. Therefore, Chapter 13 will discuss the scenarios in which recommendation technology can be used in these application domains. The specific changes that need to be made to off-the-shelf recommender systems will be discussed in this chapter together with advanced techniques such as multi-armed bandits.

1.6 Summary

This book will provide an overview of the most important classes of algorithms for recommender systems, their advantages and disadvantages, and the specific scenarios in which they are most effective. The recommendation problem will be studied in the context of different domain-specific scenarios and with different types of input information and knowledge bases. As this book will show, the recommendation problem is a rich one, and has many different manifestations depending on the nature of the input data and the scenario at hand. Furthermore, the relative effectiveness of different algorithms may vary with the specific problem setting. These trade-offs will also be explored by this book. In many cases, hybrid systems can be developed, which exploit these trade-offs effectively.

A number of advanced topics, such as attack models, group recommender systems, multi-criteria systems, active learning systems, will be studied in later chapters of this book. We will also explore a number of specific applications, such as news recommendations and computational advertising. It is hoped that this book will provide a comprehensive overview and understanding of the different scenarios that arise in the field of recommender systems.

1.7 Bibliographic Notes

Recommender systems became increasingly popular in the mid-nineties, as recommendation systems such as *GroupLens* [501] were developed. Since then, this topic has been explored extensively in the context of a wide variety of models such as collaborative systems, content-based systems, and knowledge-based systems. Detailed surveys and books on the topic may be found in [5, 46, 88, 275, 291, 307, 364, 378, 505, 529, 570]. Among these, the work in [5] is a very well written survey, which provides an excellent overview of the basic ideas. More recent surveys may be found in [88, 378, 570]. A survey of the use of non-traditional sources of information for recommendations, such as social, temporal, side information, or contextual data, is provided in [544]. A recent classification of various facets of recommender system research may be found in [462]. An excellent introductory book may be found in [275], whereas a detailed handbook [505] discusses various aspects of recommender systems in detail.

The problem of collaborative filtering with incomplete ratings matrices is closely related to the traditional literature on missing data analysis [362], although the two fields have often been studied independently. The earliest user-based collaborative filtering models were studied in [33, 98, 501, 540]. User-based methods utilize the ratings of *similar* users on the *same* item in order to make predictions. While such methods were initially quite popular, they are not easily scalable and sometimes inaccurate. Subsequently, item-based methods [181, 360, 524] were proposed, which compute predicted ratings as a function of the ratings of the *same* user on *similar* items. Another popular approach for making recommendations is the use of latent factor models. The earliest works in latent factor models independently appear in the contexts of recommendation [525] and missing value analysis [24]. Eventually, these methods were rediscovered as the most effective class of methods for performing recommendations [252, 309, 313, 500, 517]. Aside from their use in factor-based models, dimensionality reduction methods are also used to reduce the dimensionality of the ratings matrix to improve the *efficiency* of the user-to-user or item-to-item similarity in the reduced space [228, 525]. However, the work on missing data analysis is just as relevant to the recommendation literature. Other relevant models for collaborative filtering include the use of data mining models such as clustering [167, 360, 608], classification, or association pattern mining [524]. Sparsity is a major problem is such systems, and various graph-based systems have been designed to alleviate the problem of sparsity [33, 204, 647].

Content-based methods are closely related to the information retrieval literature [144, 364, 400], in which similarity retrieval methods are used in the recommendation process. Text classification methods are also particularly useful in the recommendation process. A detailed discussion on various text classification methods may be found in [22]. Some of the earliest works on content-based recommendations are found in [60, 69]. The general survey in [5] also discusses content-based recommendations quite extensively.

There are many cases in which collaborative and content-based methods are not useful in obtaining meaningful recommendations because of the high degree of complexity and constraints in the item space. In such cases, knowledge-based recommender systems [116] are particularly useful. Demographic recommender systems are discussed in [320, 475, 508], whereas utility-based recommender systems are discussed in [239]. An excellent survey on explanations in recommender systems is provided in [598].

Different recommender systems are more effective in different types of settings. The evaluation [246] of recommender systems is important in order to judge the effectiveness of different algorithms. A detailed discussion of evaluation methods may also be found in [538]. Hybrid systems [117] can combine various recommender systems to obtain more effective results. Furthermore, ensemble methods can also combine algorithms of the same type to obtain more effective results. The top entries of the Netflix Prize contest, such as "*The Ensemble*" [704] and "*Bellkor's Pragmatic Chaos*," [311] were both ensemble methods.

Recommender systems require specialized methods to make them more effective in a wide variety of scenarios. A major problem in the effective use of such systems is the *cold-start* problem, in which a sufficient number of ratings is not available at the beginning of the recommendation process. Therefore, specialized methods are often used to address this problem [533]. In many cases, the context of the recommendation, such as the location, time, or social information, can significantly improve the recommendation process [7]. Each of these different types of context has also been studied individually as a separate area of recommender systems. Temporally-aware recommender systems have been studied in [310], whereas location-aware recommender systems have been discussed in [26]. The social context is particularly diverse because it allows for a wide variety of problem settings. One can either recommend nodes or links in social networks, or one can recommend products with the help of social cues. The first of these settings is closely related to the domain of social network analysis [656]. Each of the traditional problems of ranking, node classification, and link prediction [22, 656] can be viewed as a structural recommendation problem in social networks. Furthermore, these forms of recommendation are useful beyond the social network setting. Interestingly, methods such as link prediction can also be used for traditional recommendation by transforming the user-item interactions into a bipartite graph structure [261]. A different form of social recommendation is the case where social cues are used for performing recommendations [588]. The social network structure can also be directly used in the context of viral marketing applications [297].

Since recommender systems often help the sale of products, the sellers of those products or their competitors have significant motivations to attack recommender systems by manipulating the ratings. In such cases, the recommendations are unlikely to be of high quality, and therefore *untrustworthy*. In recent years, a significant amount of effort has been devoted to the design of trustworthy recommender systems [444]. Various group recommender systems are discussed in [45, 271, 272, 407, 408, 412, 413, 415, 653]. Multi-criteria recommender systems are discussed in [11, 398, 604]. Active learning methods are discussed in [513]. A general discussion of privacy-preservation methods may be found in [20]. The earliest studies on the topic of privacy-preserving recommendations were presented in [133, 451, 484, 485, 667]. Privacy continues to be a significant challenge to such systems because of the high dimensional nature of the data. It has been shown in [30, 451] how the dimensionality can be leveraged to make privacy attacks on different types of data sets.

1.8 Exercises

1. Explain why unary ratings are significantly different from other types of ratings in the design of recommender systems.

2. Discuss cases in which content-based recommendations will not perform as well as ratings-based collaborative filtering.

3. Suppose you set up a system, where a guided visual interface is used in order to determine the product of interest to a customer. What category of recommender system does this case fall into?

4. Discuss a scenario in which location plays an important role in the recommendation process.

5. The chapter mentions the fact that collaborative filtering can be viewed as a generalization of the classification problem. Discuss a simple method to generalize classification algorithms to collaborative filtering. Explain why it is difficult to use such methods in the context of sparse ratings matrices.

6. Suppose that you had a recommender system that could predict raw ratings. How would you use it to design a top-k recommender system? Discuss the computational complexity of such a system in terms of the number of applications of the base prediction algorithm. Under what circumstances would such an approach become impractical?