

Chapter 2

Preliminaries

This chapter is organized as follows: First we present the two basic filtering techniques for recommender systems: collaborative and content-based filtering. Afterwards, in Section 2.2, we discuss how recommender systems can be compared according to different quality aspects that are relevant for the application. We describe the different types of evaluation procedures commonly found in the recommender system literature and their respective application domains. Finally, in Section 2.3, we present current recommendation approaches based on Social Web data. We focus on tagging data and identify the different approaches to leverage tagging data in recommender systems.

2.1 Basic recommendation techniques

Although recommender systems have their roots in information retrieval [Hanani et al., 2001], from the mid-1990s recommender systems have become an independent research area of their own, see [Adomavicius and Tuzhilin, 2005; Jannach et al., 2010] or [Ricci et al., 2011a] for recent overviews. Commonly, recommender systems are classified into the categories collaborative filtering, content-based filtering, and hybrid approaches [Melville and Sindhvani, 2010].

Collaborative filtering approaches exploit the wisdom of the crowd and recommend items based on the similarity of tastes or preferences of a larger user community. *Content-based* approaches, on the other hand, recommend items by analyzing their features to identify those items that are similar to the ones that the user preferred in the past. Besides to that, *knowledge-based* recommender systems exist in the literature (see, for example, [Felfernig and Burke, 2008]), which rely on explicit user requirements and some form of means-ends knowledge to match the user's needs with item characteristics, but are not covered in this thesis. In order to benefit from the advantages of the different main approaches, *hybrid* recommender systems try to combine different algorithms and exploit information from various knowledge sources. Studies have shown that for example hybrids which combine content-based and collaborative filtering can lead to more accurate predictions than pure collaborative filtering or content-based recommenders [Balabanovic and Shoham, 1997b; Melville et al., 2002].

2.1.1 Notation and symbols

In the following, we will introduce a notation and symbols for defining the recommendation problem more precisely. Our subsequent definition of the recommendation problem is based on the definition of [Adomavicius and Tuzhilin, 2005].

Let $U = \{u_1, \dots, u_n\}$ be the set of users and let $I = \{i_1, \dots, i_m\}$ be the set of items that can be recommended to the users. Furthermore we assume that $\hat{r} : U \times I \rightarrow S$ is a utility function which measures the usefulness $\hat{r}_{u,i}$ of item i to user u and returns a ranking in a totally ordered set S consisting of real numbers or nonnegative integers. Note that in the recommender system literature the utility of an item is usually represented by a *rating* value which stands for the degree a particular user likes a given item. Then, according to [Adomavicius and Tuzhilin, 2005], the recommendation problem consists

of selecting for each user $u \in U$ a not-yet-rated item $i'_u \in I$ that maximizes the utility function \hat{r} :

$$\forall u \in U, \quad i'_u = \arg \max_{i \in I} \hat{r}_{u,i} \quad (2.1)$$

Since the utility function is used to *predict* a user's interest in a particular item, we will regard \hat{r} as the prediction function in the following. Therefore, $\hat{r}_{u,i}$ stands for the predicted rating value of user u for item i . In this context, we call user u and item i *target user* and *target item* respectively. Together they build the user and item pair for which a rating prediction is made. The target user is sometimes also called the *active user*. Note that the prediction function is usually estimated (learned) in many different ways, e.g., by using methods from machine learning. The basic underlying assumption is that user ratings from the past must also be predictive of the future.

The customer ratings can be organized in a $n \times m$ rating matrix R where $r_{u,i}$ represents the rating value of user u for item i , see Figure 2.1. Note that in practice the rating matrix R is often very sparse because usually users only provide very few ratings. We use \bar{r}_u and \bar{r}_i to denote the average rating value of user u and item i respectively. The ratings from a given user u can be viewed as an incomplete rating vector \vec{r}_u . Analogously, \vec{r}_i represents the incomplete rating vector which holds the rating values assigned to the item i .

	Items					
	1	2	...	i	...	m
1	4					
2		5		4		3
⋮						
u	3	1		5		5
⋮						
n	4	5		5		4

Figure 2.1: The $n \times m$ rating matrix R where $r_{u,i}$ corresponds to the rating value (5) of user u for item i . In this example, the rating values in the user-item rating-matrix range from 1 (strongly dislike) to 5 (strongly like). The main task of a recommender is to predict the missing rating values.

2.1.2 Collaborative filtering

From the mentioned categories above, collaborative filtering is considered to be one of the most successful and promising technologies in practice [Goldberg et al., 1992; Konstan et al., 1997; Sarwar et al., 2000; Adomavicius and Tuzhilin, 2005; Jannach et al., 2010].

We will briefly describe the basic idea of collaborative filtering with the aid of an example. Table 2.2 shows an example user-item rating matrix. For simplicity, we only use a binary “Like/Dislike” rating scale. The task is to predict whether *Alice* will enjoy the movie *Scarface* or not.

	Heat (1995)	Scarface (1983)	Amélie (2001)	Eat Pray Love (2010)
Alice	Dislike	?	Like	Like
User2	Like		Dislike	Dislike
User3	Dislike	Dislike	Like	
User4		Dislike	Like	Like

Figure 2.2: Example user-item rating matrix.

In Table 2.2 we see that the users *User3* and *User4* are “similar” to *Alice* with regard to the provided ratings. Similar users are often referred to as peer users or nearest neighbors in the literature. Since both users provided a *Dislike* statement for the target movie *Scarface*, a collaborative filtering technique

will predict that *Alice* will also dislike this movie. Therefore, the movie *Scarface* is not recommended to *Alice*. Thus, collaborative filtering techniques exploit the user preferences provided by the community, that is, the wisdom of the crowd, to make a recommendation.

Memory-based vs. model-based

Collaborative filtering techniques can be further classified as memory-based or model-based [Su and Khoshgoftaar, 2009; Jannach et al., 2010]. In a *memory-based* approach the whole user-item rating matrix is kept in memory and directly used for computing predictions. In Chapter 3 a memory-based implementation is presented for our tag recommender algorithm LocalRank. In a *model-based* approach, on the other hand, the user-item rating matrix is used as input to *learn* a prediction model which is then used at run-time to make recommendations. Such a model can, for example, be the result of a data mining or machine learning algorithm. In Chapter 4, for example, a model-based method using Support Vector Machines (SVM) is introduced. For a comprehensive survey of memory-based and model-based collaborative filtering techniques the reader is referred to [Su and Khoshgoftaar, 2009].

Recommendation scheme

A traditional collaborative filtering algorithm takes a user-item rating matrix as the only input and provides a prediction function that estimates a preference $\hat{r}_{u,i}$ of user u for item i the user has not experienced before. These prediction values can then be used to compute “top- n ” recommendation lists.

Usually a user’s feeling about an item is encoded as a rating value on a seven-point or ten-point Likert response scale which ranges from “strongly like” to “strongly dislike”. Note that user preferences¹, which build the basis for the system’s representation of the user’s preferences, interests, or characteristics, can be acquired explicitly or implicitly [Miller et al., 2004]. Explicitly acquired preference values represent explicit statements by the users, whereas implicitly acquired preference values are derived automatically, e.g., by monitoring user behavior. Product ratings provided by users on Web sites such as MovieLens² or Amazon.com are examples for explicit ratings, whereas product views or the time spent on a Web page are examples for implicit ratings.

Recommender system research in the past has mainly focused on the task of exploiting the user-item rating matrix R to learn a prediction function that accurately estimates a user’s real preference values. In Section 2.2.1, we present accuracy metrics that can be used to measure the quality of the prediction function of a recommender.

In collaborative filtering, the prediction value $\hat{r}_{u,i}$ is calculated by exploiting the preferences of a larger user community, that is, the wisdom of the crowd. A basic collaborative filtering recommendation scheme for computing a prediction could work as follows:

1. *Neighborhood formation*: First, the neighbors of the target user u are determined. The neighbors represent a set of like-minded users who share similar tastes with the target user.
2. *Neighborhood selection*: Second, from the set of all neighbors the k nearest neighbors of the target user u are selected. Note that only neighbors are taken into account who provided a rating for the target item i .
3. *Aggregation of ratings*: Lastly, the nearest neighbors’ ratings for the target item are aggregated in the final prediction value $\hat{r}_{u,i}$. Aggregation can, for example, be accomplished by building the average rating value over the neighbors’ ratings for the target item.

If these steps are repeated for all items the user has not seen before, a top- n recommendation list can be generated by ranking the products according to their estimated rating values. It is important to recall that the basic underlying assumption of this recommendation scheme is that users who shared similar tastes in the past, will share similar tastes in the future. Next, we will discuss these steps in more detail.

¹Throughout this work, we will use the terms “rating” and “preference” interchangeably. The same holds for “item” and “product”.

²<http://www.movielens.org>

Neighborhood formation

Collaborative filtering approaches are usually based on neighborhood models due to their intuitiveness and simplicity. A user-based neighborhood model covers the relationships between users (see, for example, [Resnick et al., 1994]), whereas an item-based neighborhood model captures the relationships between items (see, for example, [Sarwar et al., 2001]). In the recommendation scheme described above the preferences from a user-based neighborhood were used to compute the final rating prediction. In order to find similar users or items, a similarity metric has to be defined. Pearson’s correlation coefficient is a commonly used measure in recommender systems to compute the similarity between two users a and b [Jannach et al., 2010] and is defined as follows:

$$\text{sim}(a, b) = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_{i \in I} (r_{b,i} - \bar{r}_b)^2}} \quad (2.2)$$

Generally speaking, the Pearson correlation coefficient measures the linear dependence between two variables and returns a value between -1 and 1 inclusive. Negative values correspond to a negative correlation (low similarity), while positive values indicate a positive correlation (high similarity). When calculating the correlation between users, the Pearson coefficient also accounts for a user’s individual rating behavior by taking the user’s average rating into account. Thus, a normalization is achieved before combining the rating vectors of user a and b . Note that the sum in Equation (2.2) only iterates over items $i \in I$ for which both users have provided a rating. However, in practice, extensions to the basic approaches are applied to improve performance because data sets which can often be found in practice are very sparse. *Default voting* describes one such extension which assumes a default rating value, e.g., the user’s average rating, for items for which one of the users has not provided an explicit rating [Breese et al., 1998].

Besides the Pearson correlation coefficient, other correlation coefficients such as cosine similarity and adjusted cosine similarity exist in the literature [Salton, 1989; Herlocker et al., 2004; Adomavicius and Tuzhilin, 2005; Jannach et al., 2010]. The pure cosine similarity measure is formally defined as follows:

$$\text{sim}(\vec{a}, \vec{b}) = \cos \angle(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|} \quad (2.3)$$

This measure computes the similarity between two rating vectors \vec{a} and \vec{b} and returns values between 0 (for totally orthogonal vectors) and 1 (for vectors pointing in the same direction), where higher values indicate a strong similarity. A drawback of this similarity measure is that, unlike the Pearson coefficient, it does not take the average rating behavior of the users into account. This motivated the introduction of the adjusted cosine similarity metric:

$$\text{sim}(a, b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}} \quad (2.4)$$

The adjusted cosine similarity metric factors out the average rating behavior \bar{r}_u of a user u and returns a similarity value between -1 and 1 inclusive, similar to the Pearson coefficient.

The interesting question here is when to use which similarity metric. Recommender system research has analyzed the effects of different similarity metrics on the prediction accuracy of a recommender algorithm. In [Herlocker et al., 1999], Herlocker et al. suggest to use the Pearson coefficient in a user-based recommendation approach. When using an item-based approach, however, the results of the study in [Sarwar et al., 2001] indicate that the adjusted cosine similarity metric outperforms both the basic cosine as well as the Pearson similarity metric regarding the quality dimension accuracy of a recommender algorithm. The effects of different similarity metrics for items were also analyzed in more recent works [Gedikli and Jannach, 2010; Ekstrand et al., 2011]. These results confirm the finding of [Sarwar et al., 2001]. Therefore, adjusted cosine similarity metric is often used for computing the proximity between items, whereas Pearson coefficient is appropriate for computing the proximity between users.

Neighborhood selection

The neighborhood size k plays a crucial role for the prediction quality of a nearest neighbor collaborative filtering algorithm [Herlocker et al., 1999; Sarwar et al., 2001]. If k is chosen too large, the risk of increased noise in the data will be high because not all of the neighbors are good “predictors”. Additionally, the time required for generating predictions increases. On the other hand, if k is chosen too small, the prediction quality may be affected because the prediction is based only on a few neighbors. A detailed discussion of this tradeoff relation can be found in [Herlocker et al., 1999; Anand and Mobasher, 2005; Jannach et al., 2010].

Different techniques exist to select a neighborhood size. For example, a similarity threshold can be provided such that only neighbors with a higher similarity are taken into account. The similarity threshold has to be selected properly. Otherwise, the neighborhood size could be too small or large leading to the problems discussed above. We suggest to use a user-dependent similarity threshold in order to account for users with different neighborhood sizes. Another possibility is to use a fixed neighborhood size k , leading to the question of which value chosen for k is the best. In [Sarwar et al., 2001], the authors analyze, amongst other parameters, the sensitivity of the parameter k for an item-based algorithm and a regression-based algorithm. Based on their empirical results, Sarwar et al. select 30 as their optimal choice of neighborhood size. Herlocker et al. report similar results and consider a neighborhood size of 20 to 50 as reasonable [Herlocker et al., 2002].

Aggregation of ratings

After the k nearest neighbors of user u are determined, their rating values for the target item i have to be combined together to form the prediction value $\hat{r}_{u,i}$. The standard prediction formula of the early collaborative filtering approach proposed by [Resnick et al., 1994] is defined in Equation (2.5).

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{n \in N} \text{sim}(u, n) * (r_{n,i} - \bar{r}_n)}{\sum_{n \in N} \text{sim}(u, n)} \quad (2.5)$$

The weighting factor $\text{sim}(u, n)$ is a measure of similarity between the target user u and one of his or her neighbors $n \in N$ and can be computed with one of the similarity metrics presented above. Note that various other calculation schemes exist for computing a rating prediction. However, one advantage of using the standard scheme is that the evaluation results are easier to compare [O’Donovan and Smyth, 2005a].

Recent collaborative filtering approaches

Over the last decade, a variety of collaborative filtering algorithms have been proposed in both academia and industry. Boosted by the 1 million dollar Netflix Prize competition³, in particular in the last few years a variety of sophisticated algorithms have been proposed, which rely, e.g., on matrix factorization, probability theory and advanced machine learning techniques [Lawrence and Urtasun, 2009; Koren, 2010]. In the following, we will summarize a small selection of recent approaches, in particular proposed by the author of this thesis.

In [Lemire and Maclachlan, 2005], Lemire and Maclachlan propose the *Slope One* family of item-based recommender algorithms. Slope One predictors have the form $f(x) = x + b$ and are based on the computation of “popularity differentials between items for users”. Their evaluation shows that the Slope One family leads despite its simplicity to relatively accurate predictions. Due to its simplicity, different implementations of the algorithm in various programming languages and frameworks are available today.

One of the main problems of collaborative filtering is the cold start problem [Hu and Pu, 2011; Liu et al., 2011], that is, how to filter items and make recommendations when the user community is small and the data set is sparse [Huang et al., 2004; Ma et al., 2011]. The problem can be divided into the subproblems new user and new item problem [Schein et al., 2002]. The new user problem deals with the question of how to compute recommendations for new users who did not provide any rating information

³<http://netflixprize.com>

yet. Analogously, the new item problem deals with the question of how to recommend new items for which no rating information yet exists.

In the line of this research, we recently presented a novel collaborative filtering approach called *RF-Rec* (rating frequency recommender) [Gedikli and Jannach, 2010d,e], which in particular improves the capability to deal with sparse data sets. In contrast to other collaborative filtering approaches the rating prediction for a user u and an item i not yet seen by u is solely based on the information about frequencies of rating values in the database. If, for example, the most frequent rating for item i in the database is 4 and user u 's most frequent rating is also 4, the algorithm will basically also predict 4 as u 's rating for i .

In [Gedikli et al., 2011a] we propose extensions to the RF-Rec approach in order to further increase the predictive accuracy by introducing schemes to weight and parameterize the components of the predictor. An evaluation on three standard test data sets reveals that the accuracy of our new schemes is higher than traditional collaborative filtering algorithms in particular on sparse data sets and on a par with a recent matrix factorization algorithm. At the same time, the key advantages of the basic scheme such as computational efficiency, scalability, simplicity and the support for incremental updates are still maintained.

In [Gedikli and Jannach, 2010a,b] we propose a collaborative filtering scheme which relies on the data mining technique association rule mining [Agrawal and Srikant, 1994] for generating recommendations. For each user a personalized rule set based on the user's neighborhood is learned using the recent IMSApriori algorithm [Kiran and Reddy, 2009]. At recommendation time these personalized rule sets are combined with the neighbors' rule sets to generate item proposals. The rule sets contain rules such as "*If Bob likes the movie Heat then he will also like the movie Casino*", i.e., if *Bob* watched and liked the movie *Heat* and did not watch the movie *Casino* before, the movie *Casino* can be recommended to *Bob*. The evaluation of the new method on common collaborative filtering data sets shows that our method outperforms both the IMSApriori recommender as well as a user-based k nearest neighbor method using the Pearson correlation coefficient. The observed improvements in predictive accuracy are particularly strong for sparse data sets.

2.1.3 Content-based recommendations

While collaborative filtering recommender systems recommend items similar users liked in the past, the task of a content-based recommender system is to recommend items that are similar to those the target user liked in the past [Pazzani and Billsus, 2007].

We illustrate the basic rationale of a content-based recommendation method with an example from the movie domain. Figure 2.3 represents an excerpt from an example movie database which also provides plot keywords for each movie⁴, i.e., an item's content description is represented by a set of plot keywords. Figure 2.4, on the other hand, shows an excerpt from the user database. The user database maps each user to one or more movies he or she watched in the past. In this example we assume that users only watched movies they like.

Movie Title	Plot Keywords			
Heat (1995)	Detective	Criminal	Thief	Gangster
Scarface (1983)	Gangster	Criminal	Drugs	Cocaine
Amélie (2001)	Love	Waitress	Garden Gnome	Happiness
Eat Pray Love (2010)	Divorce	India	Love	Inner Peace
...	...			

Figure 2.3: Movie data set with content description.

A simple content-based recommender computes recommendations for *Alice* by selecting movies *Alice* is not aware of and which are similar to those movies she watched before. Note that in this simple example similarity between movies could be defined by the number of overlapping keywords. Formally,

⁴The plot keywords were taken from the Web site IMDb.com.

User	Preference Profile
Alice	Eat Pray Love (2010), What Women Want (2000)
Bob	Scarface (1983), Carlito's Way (1993), Terminator II (1991)
...	...

Figure 2.4: User data set.

this idea is captured by the Dice coefficient [Baeza-Yates and Ribeiro-Neto, 1999]:

$$\text{sim}(a, b) = \frac{2 \times |\text{keywords}(a) \cap \text{keywords}(b)|}{|\text{keywords}(a)| + |\text{keywords}(b)|} \quad (2.6)$$

where a and b represent the items to be compared and the function *keywords* returns the corresponding set of keywords of a given item. The possible similarity values are between 0 and 1 inclusive. The unseen movie *Amélie*, for example, has one keyword (“Love”) in common with *Eat Pray Love*. The Dice coefficient returns $\frac{2}{3}$ in this case. Therefore, we can assume some degree of similarity between both movies. Since *Alice* liked *Eat Pray Love* in the past, *Amélie* is finally recommended to her.

As described above, a traditional collaborative filtering algorithm takes a user-item rating matrix as the only input, whereas a content-based recommender needs both the preferences of the target user as well as the textual description of the items to be recommended – the “content”. Note that for a content-based recommender no user community is required for generating recommendations. Still the cold start problem exists as the target user has to provide an initial list of “like” and “dislike” statements. However, in a content-based recommender new items can be incorporated easily in the recommendation process because similarity to existing items can be computed without the need for any rating data.

Note that in the example discussed above we did not take the importance of each keyword into account, that is, each keyword gets the same importance. However, it appears intuitive that keywords which appear more often in descriptions are less representative. Next we will show how more sophisticated metrics from the field of information retrieval address such problems and how they can be incorporated in a content-based recommender.

Content representation

In collaborative filtering two items are similar when different users have provided similar ratings for these items. Therefore, except the user-provided ratings, no additional information about the items is needed. On the other hand, in a content-based approach similarity is defined with respect to content. However, a uniform content representation of items is necessary in order to measure and compare similarity between items. Several ways exist to represent the content of an item. In the movie domain, for example, a movie’s content description can be represented by using a list of features such as director, actors, genre, description, and related-titles. If the items to be recommended are text documents such as Web sites or research articles no additional effort is necessary for acquiring the item content as they are per se descriptive, and the relation to information retrieval systems gets obvious. Therefore, content-based recommenders have their roots in information retrieval [Hanani et al., 2001; Melville and Sindhvani, 2010] where the task is to find items that match the users’ information needs and filter out unrelated items.

For convenience, we will assume in the following that the underlying item set consists of text documents. Note that item descriptions which are composed of different slots, such as director and actors in the movie domain, can be viewed as single text documents. One intuitive way to represent the (textual) content of items is to use a binary representation of the content, that is, a binary vector which indicates whether pre-selected words or phrases appear in the text document or not. However, this naive approach does not take the importance of each word and the length of each document into account. Due to the shortcomings of the naive approach, the TF-IDF encoding format was proposed and gained popularity in the field of information retrieval [Salton et al., 1975; Baeza-Yates and Ribeiro-Neto, 1999]. TF-IDF stands for term frequency - inverse document frequency and is used to determine the relevance of terms in documents of a document collection. As the name suggests the TF-IDF measure is composed by two frequency measures.

The idea of the term frequency measure $TF(i, j)$ is to estimate the importance of a term i in a given document j by counting the number of times a given term i appears in document j . Additionally, a normalization is possible, e.g., by dividing the absolute number of occurrences of term i in document j by the absolute number of occurrences of the most frequent word in document j . Several other schemes are possible.

On the other hand, the idea of the inverse document frequency measure $IDF(i)$ is to capture the importance of a term i in the whole set of available documents. Therefore, $IDF(i)$ can be seen as a global measure which reduces the weight of words that appear in many documents, e.g., stop-words such as “a”, “by”, or “about”, since they are usually not representative and helpful to differentiate between documents. Formally, inverse document frequency is usually computed as

$$IDF(i) = \log \frac{N}{n(i)} \quad (2.7)$$

where N is the size of the document set and $n(i)$ is the number documents in which the given term i appears. We assume that each term appears in at least one document, i.e., $n(i) \geq 1$. If $n(i) = N$ the logarithm function returns 0 indicating that term i is of no importance for discriminating between documents as it appears in all documents.

Finally, the TF-IDF measure, which represents the weight for a term i in document j , is defined as the combination of these two measures:

$$TF\text{-}IDF(i, j) = TF(i, j) * IDF(i) \quad (2.8)$$

With the help of the TF-IDF measure, text documents, or generally speaking the textual description of items, can be encoded as a TF-IDF weight vector. Note that many improvements have been made to the basic TF-IDF vector space model such as reducing the number of dimensions by performing stemming [Porter, 1997] and removing irrelevant stop words or uninformative keywords [Pazzani and Billsus, 2007]. Next we will see a content-based recommendation technique that relies on the vector space model.

Recommending similar items

The k nearest neighbor method can also be incorporated in a content-based filtering recommender to compute recommendations. For predicting a rating value $\hat{r}_{u,i}$ the k most similar items to the target item i , for which target user u provided a rating, are determined using the items’ content description represented by a TF-IDF vector. Afterwards, the user’s ratings for the most similar items can be aggregated to form the prediction value. A nearest neighbor content-based filtering approach is, for example, applied in [Billsus et al., 2000]. The authors present a content-based learning agent for wireless access to news which can lead to a reduction of bandwidth, time, and transmission costs. Preference information is collected implicitly by observing a user’s actions such as selecting or skipping a news story. The learned user models are used to compute a personalized list of news items.

Recent content-based approaches / hybrids

Pure content-based filtering methods typically match the user profiles with the content representation of items ignoring data from other users. In [Mooney and Roy, 2000], for example, Mooney and Roy present a content-based book recommender system. Each book is represented by a list of slots such as title, authors, synopses, published reviews, and customer comments. The information for each slot was collected from the Web pages at Amazon.com. Mooney and Roy use a Bayesian learning algorithm to compute a ranked list of book titles. While the authors show that their approach can produce accurate recommendations, in practice, however, hybrid approaches are often used which combine both techniques a content-based filtering method and a collaborative filtering approach [Balabanovic and Shoham, 1997a; Good et al., 1999; Soboroff and Nicholas, 1999; Popescul et al., 2001; Li and Kim, 2003]. Such a hybrid approach may help overcome the limitations of both approaches. A pure collaborative filtering approach is not capable of providing recommendations in cold-start situations [Schein et al., 2002], while a pure content-based approach is usually not as accurate as a collaborative filtering approach [Pilászy and Tikk, 2009]. A hybrid approach tries to combine the advantages of both methods [Balabanovic and Shoham,

1997b; Melville et al., 2002; Adomavicius and Tuzhilin, 2005]. In [Luo et al., 2009], for instance, the authors present a framework for personalized news video recommendation where different information sources are exploited for selecting news topics of interest to the user. Their proposed system supports an interactive topic network navigation and exploration process, and additionally it can recommend news videos of interest from a large-scale collection of news videos. The algorithm for ranking the news videos according to their importance and representativeness takes into account the news topic of a particular news video v , the broadcast time of v on a particular TV news program, the visiting times for v of all users, the rating score of v of all users, and the video quality in terms of frame resolution and video length. Their experimental evaluation of the algorithm on a collection of news videos⁵ reveals that the algorithm yields high-accuracy results.

2.2 Evaluating recommender systems

In this section we want to review how the quality of a recommender system with respect to different aspects can be assessed. In particular we want to focus on the different types of evaluation that exist in the recommender system literature. In [Shani and Gunawardana, 2011], Shani and Gunawardana identify three types of experiment:

- offline experiments (Section 2.2.1),
- user studies (Section 2.2.2), and
- online experiments (Section 2.2.3).

The typical characteristics of the different evaluation types are illustrated in Figure 2.5.

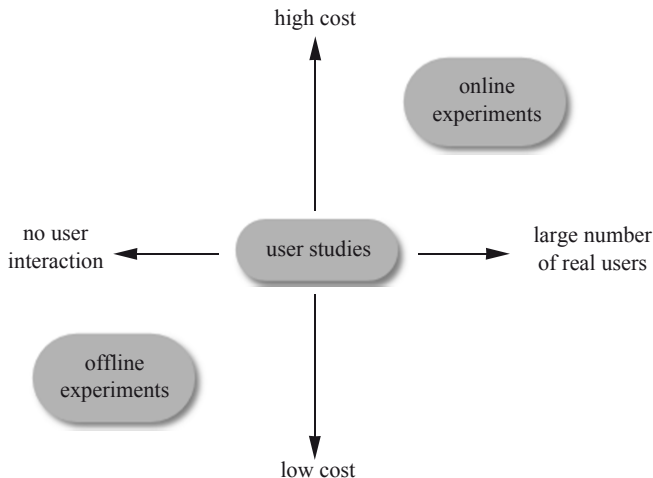


Figure 2.5: Typical characteristics of different evaluation types.

It can be seen from Figure 2.5 that offline experiments do not require user interaction at all. Therefore, they can be conducted offline at low cost and are commonly used in the literature. User studies refer to an experiment setting where a small group of users interact with the system and report their experience with the system, typically by answering a set of questions. Online experiments, on the other hand, represent large scale experiments with real users of a running system over a longer period of time. User studies and online experiments are more costly than simple offline experiments and are harder to conduct. In the following, we will explain the different evaluation types and their application areas in more detail.

⁵The data set consists of news videos captured from 3 TV news channels for more than 3 months.

2.2.1 Offline experiments

Offline experiments are easy to conduct and widely used in the literature because no interactions with real system users are required. Usually historical user transaction data is used in offline experiment settings. The freely available MovieLens data sets⁶, for example, are often used in the recommender system literature [Zhang and Pu, 2007; Yildirim and Krishnamoorthy, 2008; Symeonidis et al., 2009; Gedikli and Jannach, 2010d]. Together with the Netflix data set, which became popular due to the Netflix Prize competition⁷, they build a collection of accepted standard data sets which can be used to make the results comparable to existing approaches. Such a data set typically contains user preferences for a given item set. These user preferences are used to simulate the behavior of real users. A simple test protocol would look like follows: The user preferences are randomly assigned to a train or test set. The user preferences in the train set are used to build a user profile which then is used to predict the user's hidden item preferences in the test set. If the task to be evaluated is a prediction task, i.e., the question how accurate can the recommender algorithm make predictions, one can simply check how far the predicted value corresponds to the real preference value. Later on in this section, we will present different accuracy metrics which can be used to assess the prediction quality of a recommender.

If the used data set is publicly available and the evaluation protocol is described in detail, offline experiments can even be repeated by other researchers. In the Netflix Prize competition, for example, both the data set as well as the evaluation protocol were predefined by the organizers for the sake of reproducibility of the results.

The experiment designer has to choose the data carefully because some of the available data sets come with a bias in the data. For example, the 100k-MovieLens data set only contains users who have rated at least 20 items; the minimum number of rated items per user in the Yahoo!Movies data set⁸ is 10; the Book-Crossing data set⁹ [Ziegler et al., 2005] only contains users with at least one rating. These are examples for biases which can be found in publicly available data sets. The experiment designer should be aware of these data biases and try to correct them, e.g., by sampling the data appropriately.

In [Shani and Gunawardana, 2011], the authors point to another important drawback of offline experiments, that is, the number of questions which can be analyzed through offline experiments is limited. Usually the prediction power of recommender algorithms is analyzed in offline experiments. For more complex question types, however, user studies and online experiments are used, which will be presented in the subsequent sections.

Note that comparative offline experiments are also reported in this thesis. In Chapter 3 we come to the conclusion that our tag recommender algorithm LocalRank outperforms another algorithm because an offline experiment on a popular data set reveals that our algorithm is more efficient and has a slightly better recommendation accuracy than the baseline recommender. In Chapter 4 offline experiments are conducted to evaluate which of the tag-based algorithms performs best.

In the next section, we will describe popular evaluation metrics and procedures commonly found in offline experiment settings.

Evaluation metrics and procedures

The *Mean Absolute Error* (MAE) is a widely accepted evaluation metric in recommender system research. It can be used to measure the predictive accuracy of recommender algorithms. The MAE metric measures the average absolute deviation between the predicted rating by a recommender and a user's real (but withheld) rating. MAE can be formally defined as

$$MAE = \frac{\sum_{i=1}^N |\hat{r}_i - r_i|}{N} \quad (2.9)$$

where N is the number of tested user-item combinations. The lower the MAE value is, the better a recommender can predict a person's feelings towards an item.

⁶<http://www.grouplens.org/node/73>

⁷<http://netflixprize.com>

⁸<http://webscope.sandbox.yahoo.com>

⁹<http://www.informatik.uni-freiburg.de/~cziegler/BX>

Recommender Systems and the Social Web

Leveraging Tagging Data for Recommender Systems

Gedikli, F.

2013, XI, 112 p. 29 illus., 14 illus. in color., Softcover

ISBN: 978-3-658-01947-1