Collaborative Filtering: Matrix Completion and Session-Based Recommendation Tasks*

Dietmar Jannach[1] and Markus Zanker[2]

[1]AAU Klagenfurt, Austria
[2]Free University of Bozen-Bolzano, Italy

Abstract

This chapter provides a self-contained overview on the basics of collaborative filtering recommender systems. It covers two main classes of recommendation scenarios. In the classical matrix completion problem formulation, the task of an algorithm is to make longer-term relevance predictions given a user-item rating matrix. In session-based recommendation scenarios, the goal is to predict relevant items given a user's observed short-term behavior. From an algorithmic perspective, the chapter particularly focuses on neighborhood-based methods, which were proposed in the early days of collaborative filtering and which are still relevant today. The chapter addresses the entire life-cycle of algorithm development and also discusses libraries, datasets, and implementation aspects. Furthermore, it covers evaluation issues and reflects on today's research methodology in the field. Overall, the chapter shall serve as a starting point for readers, providing pointers to more detailed discussion of the various aspects regarding the design and evaluation of collaborative filtering recommender systems.

1 Introduction

Collaborative filtering (CF) is the predominant technical approach in the field of recommender systems in the academic literature (Jannach *et al.*, 2012). It is also utilized by large companies in productive use since clearly more than 15 years, with the system of Amazon.com being one of the most prominent early examples of wide-scale deployment (Linden *et al.*, 2003). The basic idea of this class of algorithms is to exploit the "wisdom of the crowds" and to use patterns within the collective behavior of a larger user community to determine suitable recommendations for an individual user or in the context of a given reference item.

In contrast to other possible approaches to determine recommendations, collaborative filtering techniques do not rely on content features and meta-data of the items when determining the relevance of a recommendable item for a user like content-based or knowledge-based recommendation strategies (Jannach *et al.*, 2011). In its pure form[1], collaborative-filtering relies solely on a collection of explicit or implicit *preference signals* of users towards items. Given the past preference signals of an individual user, the goal is then to determine the (current) relevance of each recommendable item by combining these individual signals with the preference patterns of the community. The corresponding output of CF algorithms usually is either a set of relevance predictions for each item or a ranked list of items.

*In: Berkovsky et al., Collaborative Recommendations: Algorithms, Practical Challenges and Applications, World Scientific, 2019

[1]A variety of additional types of data can be combined with CF approaches, e.g., data that describes the users's context or features of items. See also Chapter **??** of this book.

1.1 Historical Background

Probably the first work that used the term collaborative filtering in today's meaning in the context of recommender systems was that of Goldberg *et al.* (1992). In their *Tapestry* system, users of an experimental corporate email system could define personal *filters* (using a special query language) for incoming messages that referred to different features of the emails, e.g., the sender or its contents. "Collaborative" filters were a special class of filters, which could refer to so-called *annotations* by other users and one could therefore express that only messages should be retained that were voted positively by other users.

Soon afterwards different proposals were made of how to *automate* the task of filtering (news) items based on personal preferences and the opinions of other people. Among these proposals was the *GroupLens* system from Resnick *et al.* (1994), which proposed the comparably simple heuristic that users who shared similar preferences in the past can be exploited to predict a relevance score (rating) for incoming *netnews* messages (i.e., a user's *nearest neighbors*). While during the past two decades hundreds of different sophisticated algorithms were proposed for the rating prediction task, the problem formalization and the basic heuristic underlying this early work has influenced academic research in the field up to today.

1.2 Collaborative Filtering as a Matrix Completion Task

In Resnick *et al.* (1994), the recommendation problem is considered one of *matrix completion* (or "matrix filling" as termed in the original work). The input is a matrix where rows and columns represent users and items, respectively, and the cells of the matrix are the known preference statements (ratings) for user-item pairs.

Table 1 shows an example matrix where five users (*u1* to *u5*) rated five items (*i1* to *i5*). The recommendable items for user *u1* are items *i4* and *i5* since we assume that the *u1* already knows the other items. The question is now if we should recommend *i4* and *i5* at all, and if so, in which order the items should be recommended.

Table 1: User-item Rating Matrix.

	i1	**i2**	**i3**	**i4**	**i5**
u1	3	4	3	?	?
u2		4	3		5
u3		1	3	1	
u4	4		3	2	3
u5	3		3	2	

While the ultimate computational task in many applications is to filter and rank the items, many matrix completion approaches solve this indirectly by estimating a relevance score (or, in this case, predicting a rating) for each unknown entry in the cell. While the ranking is determined by these scores, items that do not surpass a minimum threshold need to be filtered.

Over the years, a huge variety of algorithmic approaches has been proposed to accurately predict the missing matrix values, and many of the more advanced ones will be discussed in Chapter **??** of this book. In this chapter, we will mainly focus on early and comparably simple algorithms, including the one proposed by Resnick *et al.* (1994), which is based on a nearest-neighbor scheme.

Abstracting the recommendation problem to a matrix completion task has different advantages from an academic perspective, as discussed in Jannach and Adomavicius (2016). The computational task is very well defined and the generic nature of the problem formulation allows researchers to design algorithms that are not specific to a certain application domain. Furthermore, different mathematical concepts for data analysis or noise reduction, including principal component analysis

or singular value decomposition, can be directly applied on the given data. Finally, over the years, a number of public datasets have become available and agreed-upon evaluation procedures were established in the community (see Section 4 in this chapter). These developments, together with the Netflix prize competition[2], geared research efforts on the basis of a matrix completion problem formulation during the last decade. Therefore, most of the chapters in this book will focus on these algorithmic approaches addressing this problem formulation. Nonetheless, as pointed out, e.g., in Jannach *et al.* (2016b) and Jannach and Adomavicius (2016), there are a number of aspects of practical problems for which matrix completion is not the best problem abstraction. Consequently, we will discuss an alternative problem formulation later in Section 1.4.

1.3 Basic Algorithms for Matrix Completion

In this section, we will review two basic collaborative filtering algorithms. Both of them are called "memory-based" (in contrast to "model-based" ones) because their recommendations are not based on learning an abstract representation of the data in a pre-processing step. Instead, they load the existing preference signals of the community into memory and implement neighborhood-based strategies to determine suitable recommendations.

1.3.1 User-based Nearest-Neighbor Algorithms

This algorithm scheme, which is often referred to simply as "user-based CF", implements the general idea that users "*who agreed in the past will probably agree again*" (Resnick *et al.*, 1994). To make a relevance prediction under that scheme, i.e., predicting the relevance of item $i5$ (called "target item") for user $u1$ (called the "active user") in Table 1, two main steps have to be done.

1. Identify a set of N users who exhibit similar rating patterns as $u1$ (which are often called "neighbors" or "peers") and for whom a relevance signal for item $i5$ is known.

2. Given this set of N similar users and their ratings for item $i5$, combine their ratings to predict the relevance of $i5$ for $u1$.

Consider the following example. When looking at the rating matrix in Table 1, we can see that users $u2$ and $u4$ have rated item $i5$, and the question arises if their ratings for $i5$ would be good predictors for the unknown rating of user $u1$. The basic assumption is if the ratings of user $u1$ were highly similar to those of users $u2$ and $u4$ then these neighbor's ratings for $i5$ should be useful predictors. User $u2$ rated items $i3$ and $i4$ exactly like $u1$ and user $u4$ rated them similarly, but did not use identical values.

Now, when implementing the general idea of user-based collaborative filtering in practice, two basic design choices have to be made.

a) How do we assess the similarity of two users?

b) How do we combine the ratings of the similar users?

To answer the first question, Resnick *et al.* (1994) proposed to use Pearson's correlation coefficient as a measure to assess if users have similar tastes. Given two users a and b, their similarity is computed as follows, see also Jannach *et al.* (2011),

$$sim(a,b) = \frac{\Sigma_{p \in P}(r_{a,p} - \overline{r_a})(r_{b,p} - \overline{r_b})}{\sqrt{\Sigma_{p \in P}(r_{a,p} - \overline{r_a})^2}\sqrt{(r_{b,p} - \overline{r_b})^2}} \quad (1)$$

[2] `http://www.netflixprize.com`

where P is the set of items that were rated both by a and b, $r_{a,p}$ refers to the known rating of user a for item p, and the symbol $\overline{r_a}$ corresponds to the average rating of user a. The resulting values range between -1 and $+1$, where a value $+1$ indicates that two users have identical tastes and -1 expresses that users have opposite tastes. Values close to 0 are indicators of no or only an insignificant correlation between tastes of two users. One characteristic of the correlation coefficient worth noting is that the measure accounts for different interpretations of the rating scale by two users. Instead of comparing absolute rating values, it compares how a user's rating for an item deviated from the user's average rating value.

With respect to the second question of how to combine the relevance predictions given a set of similar users N, Resnick *et al.* (1994) proposed to use the following function $pred(a, p)$, to predict the rating of user a for item p, see also Jannach *et al.* (2011).

$$pred(a,p) = \overline{r_a} + \frac{\Sigma_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\Sigma_{b \in N} |sim(a,b)|} \tag{2}$$

The idea of this prediction function is to start with the average rating of user a and then consider for each neighbor if item p has been rated above or below the individual average rating. These rating signals of the nearest neighbors are consequently weighted with the similarity between users. Thus the ratings of more similar neighbors have a stronger influence on the derived rating predictions.

Different variations of the scheme proposed by Resnick *et al.* (1994) are possible along the following dimensions.

- Similarity function: Several standard similarity measures have been explored in the literature including cosine similarity, Spearman's rho, Jaccard index or Dice coefficient (Herlocker *et al.*, 2004), where the latter measures are only applicable in case the ratings are binary or unary. Furthermore, considering only the user's deviation from their average rating value for ordinal and continuous rating scales accounts for a consistent under- or over-biasing. In addition, the similarity function could also take the number of co-rated items into account and e.g., apply significance weighting, where the similarity of users with only few co-ratings is reduced (Breese *et al.*, 1998; Herlocker *et al.*, 1999).

- Neighborhood formation: The most common strategies are to select a fixed number of nearest neighbors or to include only those neighbors above a specific similarity threshold. Alternatively, an adaptive neighborhood formation strategy can be implemented that, for instance, could dynamically adjust the similarity threshold to keep the number of nearest neighbors within a predefined range.

- Prediction function: Its task is to aggregate the ratings of neighboring users depending on their similarities and additional characteristics in order to predict a rating value. Besides the standard weighted average like in Equation 2, also specific modifications of the aggregation function were proposed, like adjusting the importance weight of controversial items with a high rating variance or amplifying the importance of very similar users (i.e., those close to the highest similarity) (Breese *et al.*, 1998).

Which algorithm configuration works best in practice has to be determined empirically based on the specifics of the given application domain.

1.3.2 Item-based Nearest-Neighbor Algorithms

The idea of "Item-based CF" is closely related to the user-based variant already discussed with the difference that we compute similarities between items based on co-ratings by individual users. When we seek a prediction for user $u1$ and an item $i5$, we do not scan the data for users that are

similar to $u1$ but for items that are similar to $i5$ and then aggregate the ratings $u1$ gave to these similar items.

For instance, turning to our example in Table 1, we can compute the similarity between those items that were rated by user $u1$ ($i1$, $i2$, and $i3$) with item $i5$ and then again compute the prediction for $i5$ as a similarity-weighted sum of the ratings for $i1$, $i2$, and $i3$. While user-based and item-based CF are technically similar, in practice item-based CF has some advantages over the user-based method, as discussed, e.g., by Linden *et al.* (2003). Consider that there are on average more ratings (or, more generally, preference signals) per item than there are per user. The similarity computations are therefore often based on a larger number of common ratings and the similarity *models* are therefore more stable. Note that in the user-based case a few more ratings can lead to a largely different set of neighbors. Given that item similarities have a tendency to be more stable, Linden *et al.* (2003) proposed to pre-compute the item similarities in an offline process to speed up the predictions at runtime. Their pre-processing method is computationally expensive in theory, but scales well as there are typically many more customers than catalogue items. Furthermore, for customers who only purchase a few best-selling items, a sampling strategy can be applied to reduce the computational effort.

1.4 Collaborative Filtering as Session-Based Recommendation

So far, our discussion was limited to scenarios where the recommendation problem is considered a matrix completion task and item relevance predictions are independent of the current usage context, users' intents, or recent observable behavior. However, in many practical applications users can have different intents each time they visit, for instance, an e-commerce site. Such short-term or ephemeral preferences are not explicitly covered in the basic problem formulation and thus specific techniques have been developed to realize context-awareness of RS.[3] In addition, visitors of commercial websites may also not be logged in, which means that in such situations no long-term preferences are known. However, individualized recommendations can be based on click-stream data of the current user session. While several works exploit other types of preference signals than explicit ratings, one of the main assumptions of standard collaborative filtering approaches is still that there is only a single type of preference signal relating users and items. Thus, advanced techniques capable of coping with multiple different signal types or with multi-dimensional ratings are discussed in later chapters. For instance, Jannach *et al.* (2018) provide a deeper discussion of recommendation algorithms based on implicit feedback.

Next, we will sketch an alternative problem formulation for session-based recommendation, where we are interested in recommendations that suit the context of a specific user session. An in-depth discussion of sequence-aware recommender systems and a more formal characterization of the problem can be found in Quadrana *et al.* (2018).

1.4.1 Inputs and Outputs

The central input to a session-based recommender is an ordered or time-stamped list of past user actions. User actions can have different types (such as "item-view" or "purchase") and are typically associated with one of the items. The users themselves can be known, i.e. recognized returning users, or anonymous. The (user, item, action) tuples can furthermore be enhanced with additional attributes like user demographics or metadata features. The central part of the inputs can be seen as a collection of enriched clickstream data, which can be easily collected on web platforms. A main difference to the matrix completion formulation is that we can potentially observe multiple interactions of a user with the same item over time.

[3] Context-aware recommenders use additional sources of information that describe the user's current situation and will be discussed in more detail in Chapter **??** of this book.

The second part of the inputs is the context of an "active" session, which consists of an ordered list of the user's actions in the session for which a recommendation is sought for. In the literature, sometimes a differentiation between "session-based" and "session-aware" recommendations is made (Quadrana *et al.*, 2018). In "session-based" scenarios, no longer-term user history is known, e.g., because we have to deal with anonymous users. In "session-aware" scenarios, in contrast, also past sessions of the active user might be known.

The output of a session-based recommender is an ordered lists of predicted next user actions, where these actions are usually related to items, e.g., a list of items for which we predict a view event or a purchase event. In the case of "next-basket predictions", the predictions can also refer to an entire set of items that are bought together. In the general form, the output is however similar to that of a traditional "item-ranking" recommendation setup.

1.4.2 Goals and Computational Tasks

In contrast to the matrix-completion task, where the goal is usually to predict a context-independent relevance score based on past preference signals, in session-based scenarios one has to also consider the short-term intents of the user. The goal is therefore to determine a ranked list of items that are relevant given the user's actions in the current session. Since the relevance of the items can largely depend on the users current contextual situation, the computational task can therefore involve balancing the users' long term preferences and their short-term situation and goals. Figure 1 illustrates the main idea of collaborative filtering as session-based recommendation.

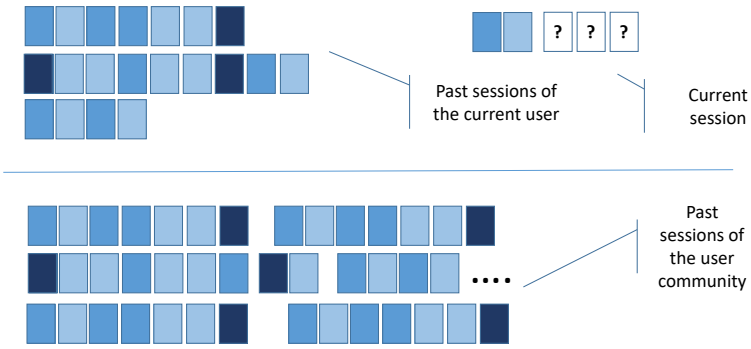

Figure 1: General principle of collaborative filtering as session-based recommendation. We are given a current user session containing actions of different types with the goal of finding suitable recommendations for the session. Both, past sessions of the individual user and the user community can be considered when determining the recommendations. Different colors in the figure indicate different types of actions.

Besides the user's intent, in session-based recommendation scenarios, the *intended purpose* of the recommendations also influences the contextual relevance of individual items. If, for instance, the goal of a recommender is to show the user alternatives for a given product of interest, items that are *similar* to those that the user has recently inspected are probably good candidates. If, in contrast, the recommender should help the user find *accessories*, a very different set of shop items becomes relevant. Finally, also the *repeated suggestion* of items that are already known to the user can be a task of for a session-based recommender, which can lead to additional computations in order to select such items and to determine the best point in time to make such recommendations.

Overall, there is no unique concept of a "good" recommendation in session-based recommendation scenarios, which also makes the evaluation of such systems more complex in academic environments as there are domain-dependent factors. With the matrix-completion problem formulation, on the other hand, several important aspects of real-world recommendation problems are not addressed at all, like short-term trends, the repeated consumption of items, or the consideration of multiple interactions of a user with an item over time.

1.5 A Nearest-Neighbor Algorithm for Session-Based Recommendation

While user-based CF and item-based CF can be considered as "canonical" approaches to the matrix completion problem that are popular baselines for algorithm comparisons, no standard baseline algorithm (and evaluation protocol) yet exists for session-based recommendation. Historically, typical technical approaches that consider the sequences of events in the recommendation process include works that rely on some form of sequential pattern mining techniques (Mobasher *et al.*, 2002), Markov models (Shani *et al.*, 2005), explicit user feedback (Zanker and Jessenitschnig, 2009b,a) and, more recently, recurrent neural networks (Hidasi *et al.*, 2016), see also Chapter **??** of this book.

An alternative to attempting to mine sequence information from the logs, which is often a computationally costly process, is to focus only on co-occurrence patterns within the different user sessions. One of the most visible examples of this approach in practice are Amazon's "Customers who bought ... also bought" recommendations. Technically, these recommendations can be implemented by considering pair-wise item co-occurrences in past transactions to predict additional items for the current shopping session.

This pairwise approach can be extended to a *session-based nearest neighbor algorithm*. The idea is to take the elements of the current user session and look for past sessions – including those by other users – who are similar to this session. We then examine which other elements appeared frequently within this set of "neighbor sessions". These elements then finally represent our recommendation candidates. In Zanker and Jessenitschnig (2009b) this similarity computation is even designed as a stepwise feature-combination approach, where the feedback categories representing different types of user actions such as "view events", "menu navigation" or explicit feedback are prioritized and low-priority feedback categories are only exploited if not sufficient recommendation candidates can be identified otherwise.

More formally, let s be the current user session, where a session is defined as an ordered list of recorded user actions. Each list entry can be a complex object describing, e.g., the respective item ID and the action type, or simply a set of item IDs if only actions of one type (e.g., "view events") are considered. Given the set S of past sessions of all users and a function $sim(s1, s2)$ that returns a similarity score for two sessions $s1$ and $s2$, compute the set N that contains those k sessions from S, which have the highest similarity score according to the function sim.

The goal is now to compute a relevance score for each recommendable item i for a given session s and a set of neighbor sessions N. We can compute this score as follows, see also Bonnin and Jannach (2014).

$$score_{\text{KNN}}(i, s) = \Sigma_{n \in N} sim(s, n) \times 1_n(i) \tag{3}$$

where $1_n(i) = 1$ if session n contains item i and 0 otherwise. The final list of recommendations is then determined as usual by sorting the recommendable items in decreasing order of the relevance score.

Although this method does not take the order of the items within a session into account, it turns out that it leads to competitive results, e.g., in the domains of next-track music recommendation, next-item recommendation in e-commerce, and next-task prediction in workflow modeling, see,

e.g., Lerche *et al.* (2016); Jannach *et al.* (2016a, 2017); Jannach and Ludewig (2017b); Ludewig and Jannach (2018).

Similar to the already mentioned user-based and item-based CF methods, different similarity functions can be applied for session-based recommendation methods. Since we typically have to cope with binary or actually unary data, i.e., an item appears or does not appear within a session, we can apply, for instance, set-based similarity measures like the Jaccard index or binary cosine similarity. Recently, different *sequence-aware similarity functions* were explored in Ludewig and Jannach (2018). The obtained results indicate that using such similarity functions in many cases further increase the prediction accuracy of neighborhood-based models. As a result, these comparably simple models often even outperform some of today's sophisticated deep learning based algorithms.

Finally, the number of neighbors is another parameter to be considered. It depends on the application domain, where, for instance, for next-track music recommendation reasonable values range from less than 10 neighbors up to hundreds of neighbors (Bonnin and Jannach, 2014; Jannach and Ludewig, 2017b).

2 Recommendation Paradigms

In this section, we will first discuss the general pros and cons of the collaborative filtering recommendation paradigm – also in comparison to other recommendation mechanisms – and will then focus on the specific advantages and limitations of neighborhood-based methods.

2.1 Pros and Cons of Collaborative Filtering

2.1.1 Existing Recommender Systems Paradigms

Recommender systems are typically categorized into collaborative filtering methods, as discussed in this book, content-based methods, knowledge-based approaches, and hybrids that are different combinations of the aforementioned paradigms (Jannach *et al.*, 2011).

The idea of content-based methods is to estimate the preference of an individual user towards the specific characteristics of items. In the movie domain, for instance, a corresponding content-based user profile could comprise to which extent a user likes action movies or an actor/actress. Thus, the profile itself is learned by analyzing the features of items that the user liked or disliked in the past.

Knowledge-based approaches also consider item features, but are usually based on explicit, formalized, and domain-dependent information about which items are a good match for a given set of user preferences. These user preferences are typically elicited in an interactive process and are specific for a given recommendation problem instance. The logic of how to match user preferences and items can, for example, be expressed in the forms of logical rules, constraints, or utility functions.

2.1.2 Comparison of CF Methods with Other Paradigms

All of these methods have their advantages and limitations, which is why various proposals to build hybrid systems were made over the years that aim at overcoming the shortcomings of individual methods.

One main advantage of collaborative filtering methods is that no knowledge about the recommendable items is required. In many application domains, including e-commerce, product catalogs can comprise tens of thousands of items, often leading to significant challenges in terms of product data management for companies. This is particularly the case when the online platform where the recommendation system is deployed is a big market place.

Pure CF-based approaches and the corresponding algorithms can furthermore be applied to a variety of domains and product categories. A large number of different algorithms was proposed over the years and a multitude of off-the-shelf implementations in different programming languages are publicly available. When compared to most knowledge-based systems, CF-based methods (and content-based ones usually as well) can learn over time when additional preference signals become available. In many knowledge-based systems, in contrast, the underlying rules have to be updated when new products with new features should be recommended.

On the downside, CF-based recommenders require the existence of a large community of users in order to be able to identify patterns in their preference relations and behavior. This requirement for a large and ideally returning user community can be particularly challenging for smaller e-commerce sites. The other main problem of CF-based systems is its limited capability of dealing with first-time users and novel items. For new or *c*old-start users, no preference signals are initially available, making it impossible to find neighbors in the user-based CF approach described above. Similar problems exist for new items that have been recently added to the item database. A huge number of academic research papers made proposals to deal with these aspects, e.g., by using initially an alternative recommendation paradigm or to recommend mostly popular items to new users, see, e.g., Bobadilla *et al.* (2012); Huang *et al.* (2004); Said *et al.* (2012a).

2.2 Pros and Cons of Nearest-Neighbor Methods

Nearest-neighbor methods in general have the advantage that they are easy to understand, implement, debug, and maintain. More sophisticated algorithms – as will be discussed in later chapters – can be challenging to implement for a typical software engineer. Furthermore, debugging unexpected outputs becomes challenging and significant engineering effort is required to deploy such algorithms in a production environment. For instance, the winning algorithms of the Netflix prize never made it into production, partly due to the involved engineering efforts (Amatriain and Basilico, 2012). In contrast to model-based approaches, which, e.g., rely on learning a prediction function from historical data in an offline process, nearest-neighbor methods are able to immediately consider new preference signals once they become available.

At least from an academic standpoint, the outputs of nearest-neighbor methods are considered to be "easy to explain", as done, e.g., in Herlocker *et al.* (2000) and a number of ways of explaining recommendations based on user-based CF methods were proposed in the literature. To which extent these academic approaches are suited in practice, is, however, still a largely open question.

Finally, nearest-neighbor methods are considered to lead to "reasonably good" " recommendations in many domains. Many more sophisticated algorithms exist which outperform nearest-neighbor methods in offline experiments in terms of predictive accuracy measure. However, it is not clear if these – often only slightly better – results can be actually noticed by users or translate into business value for the provider.

The main challenge of nearest-neighbor methods often lies in their computational complexity. When using a naive implementation, the required computation times very soon exceed the narrow time frames of online recommendation scenarios. Obviously, one cannot scan thousands or even millions of possible neighbors in real-time, whenever a new recommendation should be served to the user. Therefore, offline pre-processing, data-sampling, or parallelization techniques have to be applied to ensure the scalability of neighborhood-based methods, see, e.g., Jannach and Ludewig (2017a).

3 Practical Implementation Considerations

Recommendation systems, and collaborative filtering in particular, are a key application of data mining and machine learning at a large scale in industry. The research field of recommender

systems is traditionally closely linked to industry needs and application scenarios. Industry challenges like the famous one million dollar Netflix Prize (Bell and Koren, 2007) led to additional research momentum and the ACM conference series on recommender systems, as a result, usually attracts a high share of industry participants. Nevertheless, specific aspects of recommender system implementations traditionally only receive limited attention in academia as will be discussed next.

Scalability Amazon is one of the earliest adopters of large-scale recommendation techniques to personalize their customers' shopping experience. Thus, in an early paper on item-based collaborative filtering employed at Amazon (Linden *et al.*, 2003), the focus was put on the scalability of this neighborhood method and short response times. The importance of scalability is, for instance, also stressed in a blog post from Netflix (Amatriain and Basilico, 2012), where the strongest algorithms from the *first progress prize* could be put to practice only with significant engineering effort since they had to operate on 5 billion instead of 100 million ratings. Questions of scalability will also be discussed in Chapter **??** of this book.

Freshness and Continuous Updates In addition to the ability to scale to large amounts of data, also the freshness of the data and models is an important aspect for real-world applications. In particular, for cold-start users being able to continuously update the user model and the recommendations based on the most recent user actions within milliseconds is an important system requirement. The concept of a three-tier pipeline architecture consisting of offline, nearline and online tiers has been proposed for this purpose (Amatriain and Basilico, 2012). The offline tier performs the traditional batch processing to rebuild models at predefined intervals, the nearline tier is responsible for incremental model updates that should be very close to real-time. The online tier finally performs the actual filtering of pre-computed recommendations based on the most recent user feedback and general trends.

User Interface Francisco Martin, at that time CEO of the recommendation service provider Strands, stated in his keynote at ACM RecSys 2009 that the user interface can be much more important than the recommendation algorithm itself (Martin, 2009). Also Amatriain and Basilico (2015) mention that the user interaction design is often disregarded in the literature, even though it can have a major impact in practical systems. A recent survey of existing research on user interaction aspects of recommender systems can be found in Jugovac and Jannach (2017); aspects of user-centric evaluation approaches for recommender systems are also discussed in depth in Knijnenburg *et al.* (2012); Knijnenburg and Willemsen (2015); Pu *et al.* (2011)

Data Integration When it comes to interacting with users, almost everything that is on display at a commercial site can in principle be subject to personalization (Amatriain and Basilico, 2012). This, as a result, means that multiple categories of data sources can be considered as a potential input to algorithms, which in turn leads to a need for engineering the most appropriate features. In addition, it becomes more and more visible that explicit item ratings, as used in the context of the Netflix prize, are not the most helpful source to predict the users' next actions, e.g., because there can be a gap between what items users rate highly and which items they actually consume. Therefore, many different types of implicit feedback signals about their behavior, transactions and social connections can be exploited, as discussed in Jannach *et al.* (2018). A particular challenge in that context is that there are not only various potential types of signals but that there can be huge amounts of data to be processed. In Basilico (2013), the author for example mentions that Netflix processes over 100 billion events per day.

Business metrics Finally, when it comes to assessing the quality or value of the recommendation functionality, obviously the well-known RMSE has to be seen as just one proxy measure for the recommendation quality, but in reality more encompassing *business metrics* have to be used in practice. For instance, click-through rates, choice time, user engagement, or low churn rates are measured as success indicators in practical applications (Gomez-Uribe and Hunt, 2015).

Frameworks and Libraries From an engineering perspective, substantial progress was made in the last years and developers can today build their solutions based on several industry-strength frameworks and infrastructures that were not around a decade ago. Examples of such frameworks are the Apache PredictionIO ML server[4] or the Tensorflow[5] open source library for numeric computations using data flow graphs that has been successfully used for building deep learning and factorization-based algorithms. From a more research oriented perspective, a variety of mature open-source recommendation libraries on different language platforms are available today as well, like LensKit (Ekstrand *et al.*, 2011) for Java[6], MyMediaLite (Gantner *et al.*, 2011) for the .NET platform[7], different ML libraries such as scikit-learn (Pedregosa *et al.*, 2011) for Phyton or the *rrecsys* package (Çoba *et al.*, 2017a,b) for R[8]. Public datasets and libraries will be discussed in more depth in Chapter **??** of this book.

4 Practical Evaluation Considerations

In the following section, we will discuss common ways of evaluating collaborative filtering recommender systems. An in-depth discussion of practical challenges of evaluating recommenders can be found in Chapter **??** of this book.

4.1 General Considerations

Algorithmic approaches to build recommender systems, including collaborative filtering based ones, can in general be evaluated in different ways. The most informative way of assessing the effects of different recommendation strategies on users and the corresponding business value is to run A/B tests using websites or applications that host recommender systems. In this context, A/B testing corresponds to a randomized experiment, where the users of the system are split into two or more groups and each group is served by recommendations generated in different ways, i.e., usually the groups see different recommendations or different forms of how the recommendations are presented. At the end of the experiment period, one can then compare the effects of the different strategies, e.g., in terms of the number of clicks on the recommended items or on sales. For instance, Zanker (2012) compares two explanation strategies on a spa tourism platform or chapter 8 in Jannach *et al.* (2011) compare recommendation strategies for games on a mobile platform.

While A/B tests are the most informative measurement instrument, there are also pitfalls when designing these experiments and interpreting their results. For instance, one has to make sure in advance to have a large enough sample to ensure that any observed differences of method A compared to method B are statistically significant. Also, there can be unexpected or unconsidered periodical effects that might lead to distorted results and wrong conclusions.

At the same time, one usually cannot run a virtually infinite number of A/B tests in practice in order to evaluate hundreds of different algorithm configurations. Therefore, companies might resort to cheaper offline tests to determine those candidate configurations that most probably will

[4] https://predictionio.incubator.apache.org
[5] https://www.tensorflow.org
[6] http://lenskit.org
[7] http://www.mymedialite.net
[8] https://cran.r-project.org/web/packages/rrecsys/index.html

lead to good results in the production system (Gomez-Uribe and Hunt, 2015). We discuss such offline evaluation procedures in the next section.

4.2 Offline Evaluation

Since academic researchers usually cannot run experiments on real platforms, they commonly use methodologies to compare algorithms based on historical data. Academic research in collaborative filtering recommender systems is therefore largely dominated by such offline experimental designs.

The methodology to evaluate collaborative filtering system is mainly adopted from related research fields, in particular from the fields of information retrieval and machine learning (Herlocker *et al.*, 2004).

4.2.1 Protocols and Measures for Matrix Completion Problems

The most common procedure to evaluate recommenders is based on rating datasets. Such rating datasets contain at most one explicit or implicit preference signal for a set of users and recommendable items and can thus be considered as a typically very sparse rating matrix. The general approach is then to split the data into a training and test part, and to predict the preferences in the (held-out) test set based on the patterns that were identified in the training data. Usually, this process is repeated several times in a cross-validation process.

Rating prediction accuracy In the matrix completion task, the goal is to predict the held-out ratings and the quality of the recommendations can, for instance, be measured in terms of the average deviation of the predictions from the true (hidden) values. This measure is referred to as the Mean Absolute Error (MAE). Particularly in the last decade, researchers more often report the Root Mean Squared Error (RMSE), which penalizes larger deviations stronger than smaller ones. Details of how to calculate the measures mentioned in this section are provided, e.g., in Herlocker *et al.* (2004). Many modern collaborative filtering algorithms that will be discussed in later chapters of this book in fact try to minimize the squared error based on an offline training phase.

Looking at results that are obtained for datasets that contain movie ratings on a five-point scale, we can see that modern algorithms are able to achieve MAE values slightly below 0.7 and RMSE values that are around 0.85. The absolute values reported in the literature can however not always be directly compared even when the same dataset is used, because researchers often apply data-filtering procedures, e.g., to filter out inactive users, or use different cross-validation configurations, which leads to larger or smaller training sets, see also Said and Bellogín (2014).

Classification and ranking accuracy While rating prediction can be considered a classical machine learning (function learning) task, it is more "natural" to consider recommendation as a classification or ranking task. Accordingly, the evaluation procedures and measures from this field can be applied.

The probably most common approach according to the literature (Jannach *et al.*, 2012) is to measure and report *precision* and *recall*. Instead of rating prediction, the task of the recommender is to compute a size-restricted ranked list of recommended items for a given user. Usually, the length of the recommendation lists and correspondingly the measurement is limited to a top-10 or top-20 list of items.

The quality of such a recommendation list is then numerically quantified by considering the amount and position of "relevant" items in the top-ranked lists of all users of the test set. The precision measure counts how many items in the top-n list are relevant. With recall, we measure

how many of the relevant items actually made it into the top-n list. Since there is often a trade-off between precision and recall – longer list sizes will lead to higher recall and lower precision – researchers often report the F-measure, which is the harmonic mean of precision and recall.

The values that are reported for precision and recall, even for the same dataset, vary strongly from research paper to research paper. The main reason is that the absolute values depend on how one treats those items in the ranked lists for which no "ground truth" is known, i.e., the items for which no rating information existed in the test set. If these items with unknown ground truth are removed from the recommendation lists before determining precision and recall, the resulting values are usually very high and above 70 to 90 percent. Otherwise, if the items without unknown ground truth are kept in the list, precision and recall are usually below 10%. The low numbers for the latter case are not surprising, given that we often have thousands of recommendable items but only know the ground truth for a small subset of these items. Additional factors that influence the absolute values of the measures are the chosen list size and the number of cross-validation splits. In research works that are based on explicit (1-to-5 star) rating datasets, the outcomes are furthermore influenced by the choice of the threshold that is used to discriminate relevant from irrelevant items. Some authors only consider items to be relevant that obtained 5-stars, whereas others consider all items as relevant that have a rating that is higher than the user's average rating.

Precision and recall do not account for the position of the relevant items in the result set. Obviously, however, having a good matching proposition at position 1 is favorable over having it at a later position. Researchers therefore often use measures like the Mean Reciprocal Rank (MRR) that also consider the position of the relevant elements. More details about this and additional rank measures like NDCG or AUC etc. can be found in Herlocker *et al.* (2004).

Beyond-accuracy measures In recent years, researchers investigated a number of quality measures beyond prediction or ranking accuracy. Specifically, the question of recommendation *diversity* was analysed in depth in many works. Being able to produce a set of diverse recommendations is important in many application domains, since the recommendation of many items that are very similar (e.g., movies of a certain series) can be of limited value for the user. The diversity of a recommendation list can be quantified in different ways, e.g., based on the pairwise similarity of the items in terms of their features.

Other possibly desirable features of a recommendation algorithm can be to not only recommending popular items or, more generally, not focusing too much on a small set of items that is recommended to everyone. Furthermore, depending on the application domain, it can be desirable that the algorithm points the user to items outside of his or her usual taste. This is often considered as quality factors like *novelty* and *serendipity* (Castells *et al.*, 2015).

In most cases, considering additional quality factors comes at the price of reduced accuracy values. To be able to judge the quality of recommendations in many domains therefore requires an understanding of the specifics of the domain and a corresponding algorithmic approach to balance the often competing quality goals.

4.2.2 Protocols and Measures for Session-Based Recommendation

When evaluating session-based algorithms offline, the general principles apply as when evaluating based on rating datasets. The existing datasets, which in this case contain user action logs instead of item ratings, have to be split into training and test sets and the main computational task is to predict the hidden actions.

From the computational perspective this is however slightly different. First, as described above, rating prediction is not meaningful in this setting and our goal is usually to predict the next user action(s), e.g., the next item-view event. Second, in session-based recommendation we are provided with additional information about the user's most recent actions, which have to be considered in the computations.

In Jannach *et al.* (2015b), the authors proposed a general evaluation scheme for session-based recommendation. The overall idea is visualized in Figure 2. In a first step, the sessions of each user are split into a training and a test part, e.g., by considering the most recent 20% of the sessions of a user as test sessions. The training data represents the inputs for algorithms to learn long-term preference models. The sessions in the test set are then evaluated individually according to the given prediction task, which could be, e.g., to predict the purchase event in the session. To avoid random effects, one can repeatedly apply the protocol using, e.g., a random subsampling method.

The proposed protocol has two special parameters that can be set for an evaluation. First, there is a parameter v that describes how many items of the current session should be revealed to the algorithm when making a prediction. By varying this parameter, one can measure how quickly different algorithms are able to adapt their recommendations to the user's current goal. With the second parameter p, we can determine how many sessions that *precede* the current user session should be revealed to the session-based algorithm. By changing this parameter, we can test to which extent algorithms are able to leverage the information in these sessions, e.g., by reminding users of items that they have inspected on previous days.

While the protocol was designed for e-commerce scenarios, it is not limited to this domain. Furthermore, it can also be used in case of anonymous sessions. In this case, the training and test datasets obviously cannot be created per user and revealing previous sessions (by changing parameter p) is not possible as well.

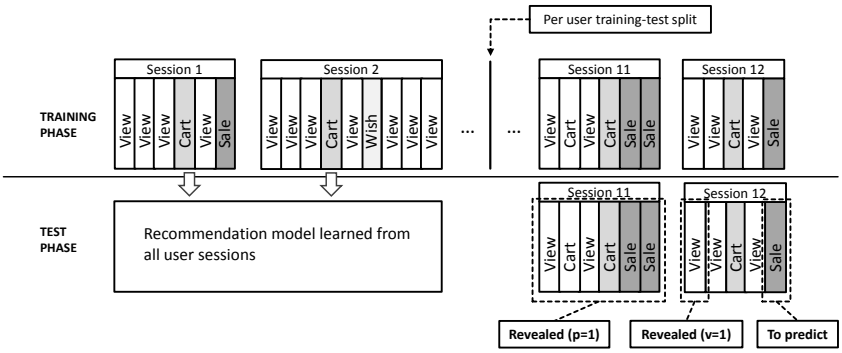

Figure 2: Session-based evaluation protocol as proposed in (Jannach *et al.*, 2015b).

Accuracy measures In general, the same classification and ranking measures described above (e.g., precision, recall, and MRR) can be applied. Depending on the chosen evaluation setup, sometimes only one item (the immediate next user action) is the relevant one and precision and recall are proportional in this case.

Since we have user actions of different types, it can furthermore be meaningful to define new measures that assess an algorithm's prediction capabilities in different dimensions. In the ACM RecSys 2015 challenge[9], for example, the task was to first predict whether or not a user will make a purchase in the current session, and if so, to predict which item will be purchased. The evaluation measure used in the challenge correspondingly considered both aspects. An in-depth discussion of evaluation aspects for session-based recommenders can be found in Quadrana *et al.* (2018).

Beyond accuracy measures Depending on the domain, again other quality factors like diversity, novelty, and serendipity can be relevant for session-based recommendation scenarios. In some

[9]http://2015.recsyschallenge.com

domains it can also be important that the recommendations represent a good "continuation" for the given session. This holds in particular when the problem is to find a suitable next track to play in a music recommendation scenario. In this application, it is typically desirable that the next played tracks are similar to the last played ones, e.g., in terms of the tempo or mood. Also in the music recommendation domain, it can furthermore be important that the set of recommended tracks is coherent in itself and the transitions between the individual tracks are smooth, see also Chapter **??** of this book. A deeper discussion of beyond-accuracy measures can be found in Chapter **??**.

4.2.3 Discussion

In this section, we reviewed basic and common approaches to evaluate recommendation algorithms in offline experiments. In the academic literature, a rich variety of alternative evaluation measures and protocol variants, e.g., to simulate cold-start situations, are used. However, even though the protocols and the measures are in principle well-defined, there are many subtle differences that can have an impact on the absolute values that are measured. Comparing the results across different research papers is therefore difficult, e.g., because different versions of precision and recall are used, datasets are pre-processed, a number of different cross-validation runs are made, or because different ways of splitting the data were applied.

The main problem when applying offline evaluation protocols however often lies on the choice of the quality measure(s). Academic research is mostly focused on prediction accuracy, but in many applications it might not be clear that higher accuracy translates into practical (business) success of the application. It is therefore important to find adequate measures that are chosen in line with the purpose of the recommender system (Jannach and Adomavicius, 2016) and which represent good proxies of the true success measures (e.g., increased sales or customer retention), which often cannot be directly measured in offline experiments.

4.3 The Role of User Studies

The aforementioned offline evaluation methodology is well-accepted in the research community. Measuring the error in terms of RMSE or MAE when assessing the rating prediction capability resembles the evaluation practice in the field of machine learning, while the methodology for assessing the classification and rank accuracy is borrowed from the field of information retrieval. Nevertheless, more and more works are challenging the view that algorithmic contributions may be solely evaluated based on offline experiments (Konstan and Riedl, 2012; Jannach *et al.*, 2016b) for several reasons. First of all, recommendation techniques aim at helping users by avoiding situations of information overload and by supporting them in decision-making tasks. Moreover, in many situations recommendation functionality should simply facilitate an enjoyable interaction of a user with an information system. Thus, recommendation systems are about enhancing the user experience and the ability of a system to accurately assess users' preferences and needs. Based on these assumptions, making correct predictions or providing many good recommendations is therefore an important component in order to make users happy and to explain their satisfaction as has been proposed by Knijnenburg *et al.* (2012). However, being able to select and present items that are relevant for users is not the only component that influences user satisfaction. Experimental user studies therefore play an important role in determining how different aspects of a recommender contribute to user satisfaction and system usage (Knijnenburg and Willemsen, 2015).

In general, following the above-mentioned works, offline evaluations can only help to assess a small part of the whole picture and therefore need to be complemented with user studies. This is in particular important since a number of recent works indicate that the quality perception or business value of a recommender system does not necessarily correlate with the accuracy measures

commonly used in offline experiments (Garcin *et al.*, 2014; Jannach and Hegelich, 2009; Kirshenbaum *et al.*, 2012; Cremonesi *et al.*, 2012).

The recent study of Rossetti *et al.* (2016), for instance, shows that an algorithm ranking based on offline experiments can contradict the outcome of ranking algorithms based on user feedback in an experimental user study, where Precision@k was measured both offline and online. For these reasons, industrial leaders often employ a development pipeline that involves three steps: traditional offline experiments to identify failing approaches from the very beginning; user studies to identify and better understand promising candidates, and A/B testing in the field as the third step in the assessment cycle.

Beyond this aforementioned pragmatic approach of employing user studies as an intermediate step before field tests, further developments of the practice how user studies are conducted can be envisioned. Specifically, since recommendation systems support decision making tasks, additional mechanisms for measurement and feedback collection can be used in the future. For example, The emerging field of NeuroIS[10] employs sensors and methodologies from neuroscience in order to more accurately understand the cognitive processes of users when interacting with technology and information systems. For instance, Rook *et al.* (2018) observed that users with specific personality traits (such as an anxiety-related behavioral inhibition) tend to be more engaged when confronted with proactive recommendations and that the accuracy of provided recommendations moderates this effect, i.e., inaccurate recommendations make them ruminate about them.

In many cases, user studies also have their limitations and the obtained insights have to be interpreted with care. The common potential threats to the validity of the obtained findings include the following. The participant population, for example, which are often students, might not be representative for the general user population or too small; often, also the study participants do not face a true decision situation and, e.g., do not actually make a real purchase. Questions about their behavioral intentions (e.g., "willingness to buy") might not reflect the findings that one would obtain in a real shopping environment. Finally, some studies suggest that there are familiarity biases in the user perception and that study participants consider items as good recommendations when they already know them (Jannach *et al.*, 2015a; Kamehkhosh and Jannach, 2017).

4.4 Datasets

Public datasets are a key ingredient when performing academic research in the field recommender systems. Such datasets, like the one from the MovieLens (ML) movie rating and recommendation platform and from companies like Netflix (Amatriain and Basilico, 2015), have led to significant algorithmic improvements in the fields over the years. In combination with public libraries that implement various algorithms, they also led to a certain level of reproducibility of the obtained research results (Ekstrand *et al.*, 2011). However, the availability of datasets also heavily biases the focus of research contributions of the community (Jannach *et al.*, 2012). The MovieLens rating datasets have been available for almost two decades (Harper and Konstan, 2016) and have significantly contributed to the fact that collaborative filtering has become the dominant recommendation paradigm in academia and that movies are the most popular application domain investigated by researchers. Table 2 shows the characteristics of a few popular datasets from the movie domain.

Many other application domains for recommender systems beyond movies have been investigated over the years as well. Often, these research works are based on datasets that were published by companies in the context of research challenges and competitions. A number of comparably popular datasets are summarized in Table 3, including data from the domains of jokes, dating, music, or general e-commerce. Overall, the increasing availability of data for different domains, different product categories and different sources will in the future allow to further intensify research on cross-domain recommendations (Cantador *et al.*, 2015), on the combination of a variety

[10] See `http://www.neurois.org/` for more information.

Table 2: Characteristics of popular movie datasets.

Dataset	Users	Items	Ratings	Scale
ML 20M[a]	138,493	26,744	20,000,263	[0.5, 5]
ML Latest Small[a]	671	9,066	100,004	[0.5, 5]
MovieLens Latest[a]	259,137	39,443	24,404,096	[0.5, 5]
ML 100K[a]	943	1,682	100,000	[0.5, 5]
ML 1M[a]	6,040	3,706	1,000,209	[0.5, 5]
ML 10M[a]	69,878	10,677	10,000,054	[0.5, 5]
ML Tag Genome[a]	1,100	9,734	12,000,000	binary
MovieTweetings 2,014[b]	24,924	15,142	212,857	[1, 10]
FilmTrust[c]	1,508	2,071	35,497	[0.5, 4]
CiaoDvd[c]	7,375	99,746	278,483	[1, 5]
Douban[d]	129,490	58,541	16,830,839	[1, 5]
Netflix[e]	480,189	17,770	100,480,507	[1, 5]

Remarks and links for download

[a] age, gender, occupation, zip, timestamp, free-text tags
http://grouplens.org/datasets/movielens

[b] https://github.com/sidooms/MovieTweetings/tree/master/recsyschallenge2014

[c] trust relationships among users
https://www.librec.net/datasets.html

[d] friendship relations among users
https://www.cse.cuhk.edu.hk/irwin.king.new/pub/data/douban

[e] dataset is officially retired, but still used in publications

of user feedback sources such as different categories of implicit transaction traces (Zanker *et al.*, 2007; Zanker and Jessenitschnig, 2009b) and on the enrichment of user preferences from social relationships.

5 Summary and Further Reading

Summary and Outlook This chapter provided an introduction on the topic of collaborative filtering with a focus on neighborhood models, since these comparably simple models even today often serve as the basis for evaluating the more sophisticated methods that will be discussed in the subsequent chapters.

Due to its introductory nature and the focus on the main underlying problem of finding items that are relevant for user, many additional academic and practical aspects of collaborative filtering have not been addressed to a large extent. Additional topics and questions to consider when designing a CF-based recommendation system include the following and a number of them will be discussed in later chapters of the book.

- Historically, academic research has focused on the rating prediction problem, which however seems to be of limited value in practice, where the main goal is to determine item rankings based on implicit feedback traces as discussed in Chapter **??** of this book. Session-based recommendations, as briefly discussed in this chapter, are an important area for future research in a field which is still dominated by research based on datasets where only one user-item interaction was recorded.
- A general issue when performing offline experiments with historical data in that context is the question how accurately collaborative filtering can actually predict user tastes, given the

Table 3: Characteristics of public datasets from different domains.

Dataset	Domain	Users	Items	Ratings	Scale
Jester1[a]	jokes	73,421	100	4,136,356	[-10, 10]
Jester2[a]	jokes	59,132	140	1,761,439	[-10, 10]
Jester3[a]	jokes	50,692	140	1,728,847	[-10, 10]
Book-Crossings (BX)[b]	books	278,858	271,379	1,149,780	[1, 10]
ISL (Views-All)[c]	cigars	1,260	142	18,434	unary
www.libimseti.cz[d]	dating	135,359	168,791	17,359,346	[1, 10]
Retailrocket[e]	e-commerce	1,407,580			unary
Scholar recs[f]	academia	50	100,531		
YOW[g]	news	24	5,921	10,010	multidim.
Epinions (665K)[h]	multiple	40,163	139,738	664,824	[1, 5]
Amazon reviews[i]	multiple	6,643,669	2,441,053	34,686,770	[0.5, 5]
last.fm[j]	music	358,868	292,375	17,535,655	unary
OSDC [k]	music		1,000,000		

Remarks and links for download

[a] http://eigentaste.berkeley.edu/dataset

[b] unary, demographic
http://www2.informatik.uni-freiburg.de/~cziegler/BX

[c] search & purchases
http://isl.ifit.uni-klu.ac.at

[d] gender http://www.occamslab.com/petricek/data

[e] user sessions
https://www.kaggle.com/retailrocket/ecommerce-dataset

[f] researcher interests
http://www.comp.nus.edu.sg/~sugiyama/SchPaperRecData.html

[g] novelty, readability, etc.
https://users.soe.ucsc.edu/~yiz/papers/data/YOWStudy

[h] trust relations
http://www.trustlet.org/epinions.html

[i] textual reviews
https://snap.stanford.edu/data/web-Amazon.html

[j] play counts, demographic
http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset

[k] tags, artists, track info
https://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset

inherent noise in the data. The discussion that algorithms cannot be accurate beyond a specific point brought up the concept of a "magic barrier" that is seen as a natural lower bound for all efforts to optimize an algorithms' accuracy (Said *et al.*, 2012b).

- In many application domains, the effectiveness of a recommendation system can depend on whether the system is able to *explain* the reasons for its recommendations (Tintarev and

Masthoff, 2011; Friedrich and Zanker, 2011; Nunes and Jannach, 2017), in particular when the goal is to design fair, accountable, and transparent systems.

- When it comes to online choice situations and the decisions users make (Chen *et al.*, 2013) a myriad of different factors like position biases, decoy and framing effects (Teppan and Zanker, 2015) or the characteristics of the rating summary statistics (Çoba *et al.*, 2018) have been shown to measurably influence the choices users make, which are mostly not (yet) considered in actual algorithms.

- From a research perspective, the reproducibility of the obtained research results is still limited in many cases despite the existence of public datasets and recommendation libraries (Ekstrand *et al.*, 2011; Said and Bellogín, 2014; Beel *et al.*, 2016; Çoba and Zanker, 2017). These topics will be discussed in Chapter **??**.

- Finally, from a societal perspective, collaborative filtering mechanisms could *fracture the global village into tribes* – a point that was already raised in the original work of Resnick *et al.* (1994) – and thus exhibit a certain tendency of reaffirming users in their beliefs and creating a filter bubble around them (Pariser, 2011). Such societal implications were not discussed so far to a large extent in the research community and represent another area where research in the field has to go beyond computer science.

Further Reading There are several highly cited works on collaborative filtering that are marking milestones of the topic like the early work on user-based automated CF (Resnick *et al.*, 1994) in an application domain (i.e., netnews) that was initially mainly addressed with content-based techniques. The proposition of an item-based neighborhood (Sarwar *et al.*, 2001) and its adoption by Amazon (Linden *et al.*, 2003) was another step into a direction towards the wider adoption of CF techniques. This obviously necessitated methodological questions about the evaluation of collaborative filtering systems as they had been addressed early in this seminal paper of Herlocker *et al.* (2004). Another important milestone for the development of the topic was the Netflix challenge and the development of many variants of scalable matrix factorization techniques, where the reader is, for instance, referred to Koren and Bell (2015) and later chapters.

Another early seminal article of Herlocker *et al.* (2000) addresses issues of algorithmic transparency and accountability by providing explanations, i.e., additional information about the recommendations and how they were derived. Later, due to the maturing of the field, a first introductory textbook appeared Jannach *et al.* (2011) that provides a comprehensive reference on collaborative filtering and its relation to other recommendation paradigms like content-based and knowledge-based techniques. Since then, even more encompassing literature like the handbook on RS with chapters on advancements in collaborative filtering algorithms (Koren and Bell, 2015) and their evaluation (Gunawardana and Shani, 2015) have been published. Finally, a recent article challenges the traditional problem formulation of collaborative filtering as a matrix completion task and advocates to also consider the user interaction and the optimization of conversational moves over time into the problem formulation (Jannach *et al.*, 2016b).

References

Amatriain, X. and Basilico, J. (2012). Netflix recommendations: beyond the 5 stars (part 1), *Netflix Tech Blog* **6**, `https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429`.

Amatriain, X. and Basilico, J. (2015). Recommender systems in industry: A netflix case study, in *Recommender Systems Handbook*, 2nd edn. (Springer), pp. 385–419.

Basilico, J. (2013). Recommendation at Netflix Scale, Talk at the first Workshop on Large Scale Recommendation Systems.

Beel, J., Breitinger, C., Langer, S., Lommatzsch, A., and Gipp, B. (2016). Towards reproducibility in recommender-systems research, *User Modeling and User-Adapted Interaction* **26**, 1, pp. 69–101.

Bell, R. M. and Koren, Y. (2007). Lessons from the Netflix prize challenge, *ACM SIGKDD Explorations Newsletter* **9**, 2, pp. 75–79.

Bobadilla, J., Ortega, F., Hernando, A., and Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem, *Knowledge-Based Systems* **26**, pp. 225 – 238.

Bonnin, G. and Jannach, D. (2014). Automated generation of music playlists: Survey and experiments, *Computing Surveys* **47**, 2, pp. 26:1–26:35.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering, in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52.

Cantador, I., Fernández-Tobías, I., Berkovsky, S., and Cremonesi, P. (2015). Cross-domain recommender systems, in *Recommender Systems Handbook*, 2nd edn. (Springer), pp. 919–959.

Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and diversity in recommender systems, in *Recommender Systems Handbook*, 2nd edn. (Springer), pp. 881–918.

Chen, L., de Gemmis, M., Felfernig, A., Lops, P., Ricci, F., and Semeraro, G. (2013). Human decision making and recommender systems, *ACM Transactions on Interactive Intelligent Systems (TiiS)* **3**, 3, p. 17.

Çoba, L., Symeonidis, P., and Zanker, M. (2017a). Reproducing and prototyping recommender systems in R, in *Proceedings of the 8th Italian Information Retrieval Workshop*, pp. 84–91.

Çoba, L., Symeonidis, P., and Zanker, M. (2017b). Visual analysis of recommendation performance, in *Proceedings of the 11th ACM Conference on Recommender Systems*, pp. 362–363.

Çoba, L. and Zanker, M. (2017). Replication and reproduction in recommender systems research - evidence from a case-study with the rrecsys library, in *Proceedings 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)*, pp. 305–314.

Çoba, L., Zanker, M., Rook, L., and Symeonidis, P. (2018). Exploring users' perception of collaborative explanation styles, in *Proceedings 20th IEEE International Conference on Business Informatics (CBI)*.

Cremonesi, P., Garzotto, F., and Turrin, R. (2012). Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study, *Transactions on Interactive Intelligent Systems* **2**, 2, pp. 11:1–11:41.

Ekstrand, M. D., Ludwig, M., Konstan, J. A., and Riedl, J. T. (2011). Rethinking the recommender research ecosystem: reproducibility, openness, and LensKit, in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 133–140.

Friedrich, G. and Zanker, M. (2011). A taxonomy for generating explanations in recommender systems, *AI Magazine* **32**, 3, pp. 90–98.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2011). MyMediaLite: A free recommender system library, in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 305–308.

Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., and Huber, A. (2014). Offline and online evaluation of news recommender systems at swissinfo.ch, in *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 169–176.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry, *Communications of the ACM* **35**, 12, pp. 61–70.

Gomez-Uribe, C. A. and Hunt, N. (2015). The Netflix recommender system: Algorithms, business value, and innovation, *Transactions on Management Information Systems* **6**, 4, pp. 13:1–13:19.

Gunawardana, A. and Shani, G. (2015). Evaluating recommender systems, in *Recommender Systems Handbook*, 2nd edn. (Springer), pp. 265–308.

Harper, F. M. and Konstan, J. A. (2016). The MovieLens datasets: History and Context, *ACM Transactions on Interactive Intelligent Systems* **5**, 4, p. 19.

Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230–237.

Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations, in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pp. 241–250.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems, *Transactions on Information Systems* **22**, 1, pp. 5–53.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks, in *Proc. ICLR '16*.

Huang, Z., Chen, H., and Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, *ACM Transactions on Information Systems* **22**, 1, pp. 116–142.

Jannach, D. and Adomavicius, G. (2016). Recommendations with a purpose, in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 7–10.

Jannach, D. and Hegelich, K. (2009). A case study on the effectiveness of recommendations in the mobile internet, in *Proceedings of the 3rd ACM Conference on Recommender Systems*, pp. 205–208.

Jannach, D., Jugovac, M., and Lerche, L. (2016a). Supporting the design of machine learning workflows with a recommendation system, *ACM Transactions on Interactive Intelligent Systems* **6**, 1.

Jannach, D., Lerche, L., and Jugovac, M. (2015a). Item familiarity as a possible confounding factor in user-centric recommender systems evaluation, *i-com Journal of Interactive Media* **14**, 1, pp. 29–39.

Jannach, D., Lerche, L., Kamehkhosh, I., and Jugovac, M. (2015b). What recommenders recommend: an analysis of recommendation biases and possible countermeasures, *User Modeling and User-Adapted Interaction* **25**, 5, pp. 427–491.

Jannach, D., Lerche, L., and Zanker, M. (2018). Recommending based on implicit feedback, in *Social Information Access - Systems and Technologies* (Springer), pp. 510–569.

Jannach, D. and Ludewig, M. (2017a). Determining characteristics of successful recommendations from log data – a case study, in *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pp. 1643–1648.

Jannach, D. and Ludewig, M. (2017b). When recurrent neural networks meet the neighborhood for session-based recommendation, in *Proceedings of the 11th ACM Conference on Recommender Systems*, pp. 306–310.

Jannach, D., Ludewig, M., and Lerche, L. (2017). Session-based item recommendation in e-commerce: On short-term intents, reminders, trends, and discounts, *User-Modeling and User-Adapted Interaction* **27**, 3–5, pp. 351–392.

Jannach, D., Resnick, P., Tuzhilin, A., and Zanker, M. (2016b). Recommender systems — beyond matrix completion, *Communications of the ACM* **59**, 11, pp. 94–102.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2011). *Recommender Systems – An Introduction* (Cambridge University Press).

Jannach, D., Zanker, M., Ge, M., and Gröning, M. (2012). Recommender systems in computer science and information systems - a landscape of research, in *Proceedings of the 13th International Conference on E-Commerce and Web Technologies*, pp. 76–87.

Jugovac, M. and Jannach, D. (2017). Interacting with recommenders – overview and research directions, *ACM Transactions on Intelligent Interactive Systems* **7**, p. 10.

Kamehkhosh, I. and Jannach, D. (2017). User perception of next-track music recommendations, in *Proceedings of the 25th Conference on User Modeling Adaptation and Personalization*, pp. 113–121.

Kirshenbaum, E., Forman, G., and Dugan, M. (2012). A live comparison of methods for personalized article recommendation at Forbes.com, in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pp. 51–66.

Knijnenburg, B. P. and Willemsen, M. C. (2015). Evaluating recommender systems with user experiments, in *Recommender Systems Handbook* (Springer), pp. 309–352.

Knijnenburg, B. P., Willemsen, M. C., Gantner, Z., Soncu, H., and Newell, C. (2012). Explaining the user experience of recommender systems, *User Modeling and User-Adapted Interaction* **22**, 4-5, pp. 441–504.

Konstan, J. A. and Riedl, J. (2012). Recommender systems: from algorithms to user experience, *User Modeling and User-Adapted Interaction* **22**, 1, pp. 101–123.

Koren, Y. and Bell, R. (2015). Advances in collaborative filtering, in *Recommender Systems Handbook*, 2nd edn. (Springer), pp. 77–118.

Lerche, L., Jannach, D., and Ludewig, M. (2016). On the value of reminders within e-commerce recommendations, in *Proceedings of the 24th Conference on User Modeling Adaptation and Personalization*, pp. 27–25.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering, *IEEE Internet Computing* **7**, 1, pp. 76–80.

Ludewig, M. and Jannach, D. (2018). Evaluation of session-based recommenation algorithms, `arXiv:1803.09587 [cs.IR]`, `https://arxiv.org/abs/1803.09587`.

Martin, F. J. (2009). RecSys '09 Industrial Keynote: Top 10 Lessons Learned Developing, Deploying and Operating Real-world Recommender Systems, in *Proceedings of the 3rd ACM Conference on Recommender Systems*, pp. 1–2.

Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2002). Using sequential and non-sequential patterns in predictive web usage mining tasks, in *Proceedings of the 2002 IEEE International Conference on Data Mining*, pp. 669–672.

Nunes, I. and Jannach, D. (2017). A systematic review and taxonomy of explanations in decision support and recommender systems, *User-Modeling and User-Adapted Interaction* **27**, 3–5, pp. 393–444.

Pariser, E. (2011). *The filter bubble: What the Internet is hiding from you* (Penguin UK).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**, pp. 2825–2830.

Pu, P., Chen, L., and Hu, R. (2011). A user-centric evaluation framework for recommender systems, in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 157–164.

Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems, *ACM Computing Surveys* `https://arxiv.org/abs/1802.08452`.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews, in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pp. 175–186.

Rook, L., Sabic, A., and Zanker, M. (2018). Reinforcement sensitivity and engagement in proactive recommendations: Experimental evidence, in *Information Systems and Neuroscience - Gmunden Retreat on NeuroIS 2017*, pp. 9–15.

Rossetti, M., Stella, F., and Zanker, M. (2016). Contrasting offline and online results when evaluating recommendation algorithms, in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 31–34.

Said, A. and Bellogín, A. (2014). Comparative recommender system evaluation: Benchmarking recommendation frameworks, in *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 129–136.

Said, A., Jain, B. J., and Albayrak, S. (2012a). Analyzing weighting schemes in collaborative filtering: Cold start, post cold start and power users, in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 2035–2040.

Said, A., Jain, B. J., Narr, S., Plumbaum, T., Albayrak, S., and Scheel, C. (2012b). Estimating the magic barrier of recommender systems: a user study, in *Proceedings of the 35th ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1061–1062.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms, in *Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295.

Shani, G., Heckerman, D., and Brafman, R. I. (2005). An MDP-Based Recommender System, *The Journal of Machine Learning Research* **6**, pp. 1265–1295.

Teppan, E. C. and Zanker, M. (2015). Decision biases in recommender systems, *Journal of Internet Commerce* **14**, 2, pp. 255–275.

Tintarev, N. and Masthoff, J. (2011). Designing and evaluating explanations for recommender systems, *Recommender Systems Handbook* , pp. 479–510.

Zanker, M. (2012). The influence of knowledgeable explanations on users' perception of a recommender system, in *Proceedings of the 6th ACM Conference on Recommender Systems* (ACM), pp. 269–272.

Zanker, M. and Jessenitschnig, M. (2009a). Case-studies on exploiting explicit customer requirements in recommender systems, *User Modeling and User-Adapted Interaction* **19**, 1-2, pp. 133–166.

Zanker, M. and Jessenitschnig, M. (2009b). Collaborative feature-combination recommender exploiting explicit and implicit user feedback, in *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing*, pp. 49–56.

Zanker, M., Jessenitschnig, M., Jannach, D., and Gordea, S. (2007). Comparing recommendation strategies in a commercial context, *IEEE Intelligent Systems* **22**, 3.