14

Recommending based on Implicit Feedback

Dietmar Jannach[a] and Lukas Lerche[a] and Markus Zanker[b]

[a]Department of Computer Science, TU Dortmund University,
44221 Dortmund, Germany
dietmar.jannach@tu-dortmund.de
lukas.lerche@tu-dortmund.de

[b]Free University of Bozen-Bolzano,
39100 Bozen, Italy
mzanker@unibz.it

Abstract. Recommender systems have shown to be valuable tools for filtering, ranking, and discovery in a variety of application domains such as e-commerce, media repositories or document-based information in general that includes the various scenarios of Social Information Access discussed in this book. One key to the success of such systems lies in the precise acquisition or estimation of the user's preferences. While general recommender systems research often relies on the existence of *explicit* preference statements for personalization, such information is often very sparse or unavailable in real-world applications. Information that allows us to assess the relevance of certain items indirectly through a user's actions and behavior (*implicit feedback*) is in contrast often available in abundance. In this chapter we categorize different types of implicit feedback and review their use in the context of recommender systems and Social Information Access applications. We then extend the categorization scheme to be suitable to recent application domains. Finally, we present state-of-the-art algorithmic approaches, discuss challenges when using implicit feedback signals in particular with respect to popularity biases, and discuss selected recent works from the literature.

Key words: Implicit User Feedback; Recommender Systems; Collaborative Filtering.

14.1 Introduction and Motivation

Recommendation is a key functionality on many modern websites and mobile applications. The task of recommendation components within applications is typically to point users to additional items of interest by ranking or filtering them according to the past preferences and the current contextual situation of these users.

Mainstream research in the field of Recommender Systems (RS) – as discussed in detail in Chapter 10 of this book [55] – was historically fueled by applications scenarios in which preference statements of users in the form of *explicit item*

ratings are available [49]. This led to the development of sophisticated algorithms that are able to very accurately predict which rating a user would probably give to a certain item. Much of the power of these algorithms is based on the existence of large datasets of historical ratings in which for each user dozens of explicit ratings exist. Since the evaluation is often only done on this historical offline data, the algorithms are optimized to accurately "post-dict" recommendations rather than to predict (which may or may not overlap with real-world performance). While there exist a number of dedicated Social Web platforms on which users can rate movies, books, restaurants or other businesses, there are also many real-world application domains in which rating matrices are very sparse or even non-existent today [42, 43]. For example, while some popular items on Amazon.com receive many ratings, most of the items in the catalog do not have any ratings. Also, in domains like friend discovery for social networks, people usually can not be rated explicitly.

When building personalized recommenders in such application domains we have to rely on *indirect ways* of assessing the interests and preferences of users by monitoring and interpreting their actions and behavior. In the research literature, these observations of a user's actions that are interpreted as *statements on the relevance of a particular item* are called *implicit feedback*. Sometimes also the term "nonintrusive" feedback is used because users are not explicitly stating their preferences, but these are derived from their observed actions. In a classic e-commerce setting, an example of a user action that might indicate a preference for an item is when the user views the detailed product description of an item or puts the item on a wish list. On media streaming platforms, the repeated consumption of a track or music video can be interpreted as an interest or preference of the user toward the track itself, the track's artist or the genre. On a social network, sharing a certain news story in a post might express the user's general interest in the topic, as further discussed in Chapter 11 of this book [63].

Implicit and explicit feedback are however not a set of boolean categories, but rather a continuum. Consider the case of a user playing a music track or sharing a news article. These actions can be interpreted as implicit feedback, i.e., the user might have a preference towards the track or the article contents. We might also infer from the user actions that he is interested in the track's artists or the topic of the news story. However, if the user (explicitly) gave the track a rating or "liked" the news article, we could also (implicitly) infer that he might be interested in the artist or topic. Therefore, when we speak of *implicit feedback*, we mean all kinds of interactions with the systems from which we can indirectly *infer* user preferences.

The amount of available implicit preference signals in reality can be huge. Today, every mouse move of a user can in theory be tracked in an online application. In the future, with the continuing development of the *Internet of Things* and users being "always-on" through mobile or wearable devices, even larger amounts of information about the users' behavior and about the objects with whom they interact with will be available.

Besides the technical challenge of efficiently processing such a constant stream of possibly large amounts of data, a number of further questions has to be addressed. These questions include, for example, which of the many types of signals should be used to build a preference profile and how to combine these signals with possibly existing explicit rating information. Furthermore, different signals might indicate a different "strength" of a preference, i.e., a purchase may count more than an item view action in an online store. Finally, implicit feedback signals are often positive-only and in addition we cannot be always sure that we interpret the signals correctly as, e.g., an online shopper can be disappointed later on with a purchase or was purchasing something for a friend.

Overall, recommendation based on implicit feedback in real-world applications is probably much more common than relying (solely) on explicit ratings, e.g., because the acquisition of ratings requires certain efforts from the user's side. A general problem of explicit ratings is that many users use ratings as a means to assess the *quality* of an item, e.g., a movie, than to express their *enjoyment*, which is probably more relevant in a recommendation scenario.[1] Recommendations based on the true user behavior might therefore in fact be more reliable than predictions that are based on explicit ratings in reality.

In this chapter we will review existing approaches and challenges of creating user profiles for recommendation based on implicit feedback. These types of preference signals are particularly common in Social Information Access scenarios discussed in this book. Think, e.g., of users who share, comment on, or tag resources in Social Web applications (Chapter 8 [35], Chapter 11 [63] and Chapter 12 of this book [8]); people who regularly check in at certain locations on location-based social networks (Chapter 16 of this book [12]); or music lovers who post their playlists and connect with other users on music streaming platforms (Chapter 11 of this book [63]). These and various other types of user actions can be used to create additional recommendations on the corresponding platforms.

Figure 14.1 shows the structural outline of this chapter. In Section 14.2, we characterize explicit and implicit feedback signals and discuss challenges when detecting and interpreting such feedback. We then give examples for implicit-feedback application domains of recommender systems in Section 14.3 and propose an extension to Kim and Oard's [85] classification scheme for implicit feedback. In Section 14.4, we review typical algorithmic approaches to deal with implicit feedback signals. Section 14.5 discusses evaluation aspects. In Section 14.6 we finally present a number of case studies and selected technical approaches from the literature.

[1] `http://finance.yahoo.com/news/netflix-wants-ditch-5-star-202428660.html`

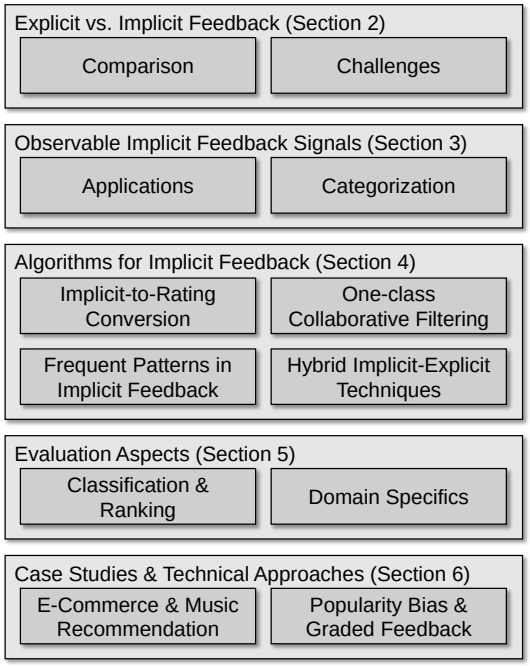

Fig. 14.1 Structural outline of the chapter

14.2 Explicit vs. Implicit Feedback

A number of different types of input data can be used in the recommendation process and the research literature typically differentiates between explicit and implicit feedback. However, as mentioned in the introduction, implicit and explicit signals are not boolean categories, but a continuum, because explicit feedback can to some degree infer implicit preferences. In the following, we will characterize both implicit and explicit feedback, as well as differentiate them from other input that is neither. Lastly, we will address challenges when implicit feedback is used in recommender systems.

Explicit feedback in general corresponds to a deliberate, unambiguous, and intentional quality assessment by a user on the performance of a system. These assessments are obviously dependent on the application domain. For instance, in the context of a recommender system this feedback is typically related to the relevance of a specific item in a given situational context.

In contrast, implicit feedback lacks this user intention to provide an opinion to the system, but it subsumes all sorts of user actions or behavior which can be exploited by a system to infer the relevance of its propositions to users, i.e., estimate the positive or negative bias towards a specific item *or* towards items with similar characteristics. Obviously, the exploitation of implicit feedback comes at the cost of the uncertainty when interpreting it, i.e. first, the implicit signal

may not always unanimously represent a positive or negative bias. For instance, viewing time can be an indicator of interest, of having problems to understand the content or can be the result of being distracted by other uncontrolled events. Second, even if the direction of an implicit feedback category is unanimous its quantification in order to aggregate it with other implicit signals and explicit preference statements adds uncertainty.

14.2.1 Explicit Feedback

The most prominent form of such feedback in the literature are user-provided *ratings*, e.g., on a 1-to-5 scale often displayed as "stars". In most settings, only one overall rating per item is available. In multi-criteria recommendation approaches, more fine-grained rating feedback regarding different quality dimensions of the items is used. The topic of rating-based collaborative filtering techniques is further discussed in Chapter 10 of this book [55].

Apart from star ratings, other common forms of explicit feedback are unary "like" or "recommend to a friend" statements as well as binary "thumbs up", "thumbs down" selections. In certain applications, we also find explicit negative user actions such as "banning" a track on a music streaming platform or blocking or hiding certain messages on a Social Web platform. Though the latter signals are unary or binary, they are not *implicit* feedback. Sometimes these aspects are confused as most implicit feedback algorithms only rely on unary or binary signals and can therefore be applied for these feedback types as well, as mentioned in the discussion of implicit feedback algorithms later on in Section 14.4.2.

Besides these directly processable preference expressions, there are other forms of explicit feedback which however require further analysis or which are application-specific. On the Social Web, users can for example express their opinions through reviews in natural language or by annotating items with tags that have a (known) positive or negative connotation, see also Chapter 13 of this book [86]. An example for an application-specific explicit feedback would be that a user of an online bookstore puts a book on a "recommended reading" list. Also adding a browser bookmark for a website *can* be an explicit statement in case the bookmark is put into a folder with a clear positive or negative connotation, e.g., "My Favorites".

The distinction between the different feedback types for these latter cases can however be a continuum and explicit statements might infer further implicit preferences. For example, any bookmarking action – independent of the fact that we potentially can unambiguously derive the users' quality assessment for the item – is never an explicit feedback signal, because the users' intention is *not* to inform the system about their preferences in the first place. Such an argument could also be raised for explicit star ratings, where the users' main intention might be to use the rating as a personal reminder for themselves or to share their experiences with other users and not state their opinion in the first place.

In addition, however, explicit rating information may be sparse as such ratings require extra work by the users who might not immediately see the benefit of specifying the preferences. Furthermore, providing an explicit rating requires a considerable amount of cognitive effort by the users and some might be challenged in expressing their preferences using a single rating on a pre-defined and often coarse scale, as reported, e.g., in the study in [124]. This study explored how different factors influence the utility of implicit relevance feedback in search systems and identified that the task complexity had a considerable impact on the users' preference to provide implicit or explicit feedback. In complex search tasks where users rarely identified fully relevant objects, implicit feedback was preferred, because the users' focus centered around the search task. Based on these findings one could hypothesize that users would prefer implicit feedback in domains where the primary task requires their full attention, such as in online shopping, while in media and entertainment domains they might be more willing to provide explicit feedback.

14.2.2 Implicit Feedback

As mentioned before, implicit feedback subsumes all sorts of user actions or behavior that were not intentionally executed in order to provide feedback on specific items or the system performance in general. However, these implicit signals can be observed and are worthwhile to exploit in order to infer a positive or negative user bias towards a specific item, towards items with specific characteristics, or towards a specific action taken by the system. Usually, one of the tasks when using implicit feedback is to find a suitable way of interpreting the feedback, for example, by mapping it onto a rating scale or by learning relative (pair-wise) preference models.

14.2.2.1 Observable User Actions. The typical types of such interpretable signals are observable user actions, e.g., when users view or purchase some items on an online store, when they select news articles of certain topics to be displayed, when they listen to a track on a music streaming portal, when they tag or bookmark a resource, or join a group on a social network. The user's navigation behavior – from category browsing to mouse and eye movements – represents another typical category of implicit feedback.

An early categorization of possible types of observable implicit feedback signals – focusing on information filtering and recommendation – can be found in [83]. This classification was later extended by Oard and Kim in [85], who identified three types of observable behavior: ***Examination***, ***Retention***, and ***Reference***. Later on, in [84], a fourth category – ***Annotation*** – was added, which in some sense unifies implicit and explicit feedback based on the types of observable behavior [49]. In the bibliographical review presented in [52], the authors introduce a fifth dimension called ***Create***, which relates, e.g., the user

activity of writing or editing an original piece of information. The five categories of implicit feedback are summarized in Table 14.1.

Table 14.1 Summary of the five types of observable behavior, adapted from Oard and Kim in [84], [85], and [52].

Category	**Examples of Observable Behavior**
Examination	Duration of viewing time, repeated consumption, selection of text parts, dwell time at specific locations in a document, purchase or subscription.
Retention	Preparation for future use by bookmarking or saving a named or annotated reference, printing, deleting.
Reference	Establishing a link between objects. Forwarding a document and replying, creating hyperlinks between documents, referencing documents.
Annotation	Mark up, rate or publish an object (includes explicit feedback).
Create	Write or edit a document, e.g. [13] or [39].

We will take another look at these five types of observable behavior later on in Section 14.3.2 after we have reviewed recent research on implicit feedback in recommender systems and related fields. We will see that the development of recommendation technology over the last two decades suggests that this classification should be extended and present a suitable extension later on in Table 14.2.

14.2.2.2 User-Action-related Indirect Preference Signals. Implicit feedback for an item can also be inferred from indirect preference signals that are based on explicit feedback (ratings) on related objects or from other user actions that are not directly related to a specific item. We use the term "preference signals" here as the user's actions usually can not be directly considered as feedback on a specific item. An explicit "like" expression for an artist on a social music platform can, for instance, be used as a positive signal for the artist's musical pieces in a music recommender system. Such types of information are usually exploited by content-based filtering recommender systems, which often rely on these forms of "indirect" preference signals.

14.2.2.3 User-Feature-related Indirect Preference Signals. User demographics, the user's *current* location, or the user's item-independent embedding in a social network are usually *not* considered as implicit feedback. Depending on the application scenario, some of these user features *can* however represent indirect preference indicators, i.e., a form of implicit feedback, if the characteristics are the results of user actions that are at least indirectly related with the recommendation targets.

For example, in a restaurant recommender, information about the user's *past* geographic location and movement profile *can* be considered as a form of implicit preference signals in case the movement profile allows us to infer a restaurant preference of a specific user without having the user explicitly "checked in" to the restaurant. An in-depth discussion of location-based recommenders can be found in Chapter 16 of this book [12]. Also, the user's connections in a social

network *can* be considered as implicit preference signals in particular when the goal is to recommend people or groups (see also Chapter 15 of this book [31]).

14.2.2.4 Discussion. As we have seen, apart from observable implicit feedback, a variety of additional preference signals can be used in the user profiling and recommendation process including in particular the users' demographics or other user characteristics that are independent of an individually recommended item. In the categorization of different feedback types, these signals are usually not considered as implicit feedback. Furthermore, user-independent, additional information about items – including information about item features or to which other items they are connected – is also not considered to be implicit feedback per se, but it might be useful in correctly interpreting implicit feedback signals such as listening or viewing actions. Similarly, contextual information about the users like the location or time when a specific explicit rating was issued, do not fall into this category of implicit feedback, but help to contextualize the collected feedback.

Overall, the distinction between explicit and implicit feedback and other types of information is not always consistent in the research literature and, as discussed, cannot be seen as a boolean categorization. However, one common aspect of all kinds of feedback, that is not explicitly meant to provide an opinion or a relevance assessment, is a set of specific challenges that will be discussed next.

14.2.3 Challenges of Using Implicit Feedback

When relying on implicit feedback, a number of challenges has to be addressed. This list is by far from being complete and we recommend to see also the discussion in [49].

14.2.3.1 Interpretation of Signal Strength. In many situations, several types of user actions have to be considered in parallel and the question on how to aggregate them turns up. Usually a uniform weighting strategy might not be appropriate. For example, in an e-commerce scenario a purchase action might be a stronger preference indicator than a repeated item visit. In addition, the different strengths of implicit feedback signals could be determined by additional post-processing steps, e.g., for identifying different degrees of friendship between users based on their observed communication patterns as done in [105].

14.2.3.2 Interpretation in Relation to Explicit Signals. Sometimes, both explicit and implicit feedback signals are available, but with different degrees of coverage of the item space. Therefore suitable ways of combining them are needed. Simple approaches in which all implicit actions are, e.g., interpreted as a "four star" rating on a five-item scale and this way transformed into explicit rating signals are popular but inappropriate as the rating database becomes "dominated" by the large amounts of implicit signals. Often the implicit feedback

"scales", e.g., visit duration, track playcounts etc., are also incompatible with the five-point scales used for explicit feedback.

14.2.3.3 Transparency. When explicit feedback is available, it might be easier for the user to understand the rationale of the provided recommendations as they, e.g., can be used in system-generated explanations. Recommendations that result from implicit feedback signals might not be that obvious or plausible for the user. For example, showing a recommendation to a user with the explanation "because you rated [movie A] with 5 stars" might be more plausible than the explanation "because you watched [movie A]", as in the latter case the user might not have liked movie A after all.

14.2.3.4 Lack of Negative Signals. Implicit feedback is often "positive-only", i.e., we only can learn positive biases from a user's interaction with an item. This lack of negative signals often means that special types of algorithms (one-class collaborative filtering) have to be applied. This also leads to challenges when applying standard evaluation measures as no ground-truth about non-relevant items is available.

14.2.3.5 Data Not Missing at Random. In most domains, implicit feedback signals for the few very popular items are prevalent while feedback for niche items can be very sparse [77]. Therefore, the distribution of feedback is skewed in a long-tail shape. Building recommendation models based on such data can easily lead to a strong popularity bias ("blockbuster effect") and a "starvation" of the niche items.

14.2.3.6 Abundance of Data. The computation of sophisticated machine learning models can be challenging on large platforms even when only explicit ratings are considered. The amount of data points to be processed, if for example every single navigation action of a user is logged, makes this problem even worse. Furthermore, given the variety of available types of data points, it is not always clear which of the many signals are the most promising ones to retain and consider in the recommendation process.

On the other hand, while implicit feedback signals have some disadvantages when compared to explicit ratings, one advantage of implicit signals is that they can be collected from all users, while (sufficient amounts of) explicit rating information might in many domains only be available from a few "heavy" users. As a result, the models that are learned solely from explicit ratings might over-represent some user groups.

14.3 Categories of Observable Implicit Feedback Signals

In this section we will review typical examples of applications from the research literature that use implicit feedback. We will then come back to the previously discussed categorization scheme for implicit feedback by Oard and Kim and

propose an extension with additional types of user actions which have become observable due to technological advancements during the last years.

14.3.1 Types of Observed Behavior in Applications

Historically, one of the various roots of today's recommender systems lies in the field of Information Filtering, an area that dates back to the 1960s under the term "Selective Dissemination of Information" [37]. The main tasks of information filtering systems typically are to identify and rank documents within larger collections based on their presumed degree of relevance given the user's profile information. Recommender systems nowadays are used in various applications domains, e.g., e-commerce, media consumption and social networks.

In the following, we will give examples of research works from the recommender systems literature to illustrate the various (new) ways of how user actions and observable behavior can be interpreted and used in different application scenarios. The review of existing works will serve as a basis of our proposal to extend the categorization scheme of [85] in Section 14.3.2.

14.3.1.1 Navigation and Browsing Behavior. Monitoring how users navigate a website or how they use a (web-based) application is a very general type of observable user actions. Several early works that focused on implicit feedback aimed at the *dynamic content adaptation* by, e.g., generating links to possibly additionally relevant content or filter the available content according to the user's preferences.

Analyzing Dwelling Times for Information Filtering. As mentioned in Section 14.2, interpreting dwelling time as implicit feedback is a challenging task. One of the earlier works in the area of personalized information filtering that tries to rely on the observation of the users' behavior, e.g., dwelling times, to infer their interests is found in [82]. The authors' specific assumption was that users of their NetNews system will spend more time on interesting items than on non-interesting ones. To verify their hypothesis, they designed a study in which users had to read news articles during a period of several weeks and provide explicit ratings for the articles. The collected data then indeed showed that reading times are good indicators for the relevance of an article and that both the length and the readability of an article only had a limited impact on reading time. The news filtering systems discussed later in [57] or [107] had similar goals and the studies confirm that relying on reading times alone can help to generate accurate recommendations in many situations. Furthermore, as mentioned in [56], too short viewing times can also be interpreted as *negative* implicit feedback and not only as *missing positive feedback*. The complexity of the interpretation of dwelling time as positive or negative feedback will be further discussed in Section 14.4.1.3.

Monitoring Navigation Actions. Before the large success of WWW search engines, a number of proposals were made to help users with finding relevant websites based on the observation of their browsing behavior. An early approach of that type relying on the user's browsing behavior to infer the user's interest is the system "Letizia" [70]. The client-side looks at the links that are followed by a user, at initiated searches, or at bookmarking activities and applies content-based heuristics to find additional relevant web pages. Other early tools that are similar to the basic idea to customize recommendations based on the users joint navigation behavior (e.g., link selection) and document content similarities are described in [6] or, with a focus on personalized recommendations, [79].

Browsing Actions. In [56], a number of additional browsing-based interest indicators besides the following of hyperlinks are mentioned, including micro-level actions like scrolling, highlighting or the visual enlargements of objects. Depending on the installed equipment on the client side, one can also try to capture the eye gazes of the users [15] or approximate them by tracking the user's mouse movements [104]. From a technical perspective, server-side logging of client-side actions can nowadays be implemented very efficiently using AJAX-based micro requests. Further user interface level actions include requesting help or explanations for an object.

Web Usage Mining. In contrast to approaches that only rely on navigation or browsing logs of individual users, web usage mining systems aim to detect usage patterns in the logs of a larger user community using, e.g., clustering or association rule mining techniques. Personalization systems like WebPersonalizer presented in [80] for example try to match the current user's most recent navigation activities with "aggregated profiles" to generate personalized website recommendations.

Discussion. The "Social Information Access" aspect is most obvious in the last category (Web Usage Mining) where the behavior of other users in the community is directly exploited to make suggestions for the current user. Nonetheless, also the other presented techniques which were partially designed for individual-user settings can in principle be extended to consider the behavior of the community, e.g., by adding collaborative features within the server-side components.

14.3.1.2 Shopping Behavior. Implicit feedback signals in e-commerce applications – and also others as mentioned below – could in principle be considered as a subclass of the *navigation and browsing behavior*. However, in a commercial context specific *semantic meanings* can be attached to some navigation actions such as viewing an item or adding it to a wishlist or to the shopping basket, while usually not all navigation actions are considered to be relevant for exploitation.

Shopping Basket Analysis. Amazon.com's *Users who bought ... also bought ...* denotion of one of their recommendation lists characterizes the main idea of such approaches quite well. The general underlying concept is to find patterns in the shopping baskets of users [74]. Often, these patterns are identified using more

general techniques like classic Association Rule Mining [2] or variations thereof, which can then be applied to make recommendations for the current user [73].

Shop Visitor Navigation Logs. Another category of recommendations on Amazon.com's site is named "Users who viewed . . . ", which expresses that also other types of user actions can be used for building user profiles on shopping sites. One difference to the above-mentioned general approaches based on navigation logs is, as said, that a purchase is a very distinctive action and one of the main business metrics to be optimized. Recent examples of works which aim to exploit the user's recent navigation behavior to predict the next shopping action include [44, 102, 111, 119] and are often based on approaches that model sequential decision processes.

Discussion. In the past, academic researchers often converted explicit rating datasets into "purchase transactions", e.g., by considering five-star ratings as purchases, because not many public datasets were available. In recent years, we see an increased rate of works which are based on real-world shop navigation logs. Academic competitions like at the 2015 ACM RecSys Challenge[2] help to fuel these types of research as they are based on publicly available real-world datasets. With the emergence of the Social Web, more and more shopping platforms allow their users to comment, review, and share their experiences on the site, and a variety of other user-related data becomes available for specific tasks like next-basket predictions.

14.3.1.3 Media Consumption Behavior. Reading news online is, as described above, a classic information filtering scenario in which implicit feedback was explored. Other types of electronic media consumption in which implicit feedback recommendation systems were employed include the recommendation of (IP) TV programs based on viewing times, video recommendations using the watching behavior or music recommendation based on listening logs.

Implicit feedback signals related to media consumption often face additional challenges. Both for music and TV shows it is not always clear who – if anyone at all – in the household is currently watching or listening. In addition, user actions like a "skip" to the next track can be context dependent and interpreting it as a general negative assessment of the previous track might be misleading.

TV-related Recommendations. Recommending based on implicit feedback in the context of TV programs was for instance explored in [26], where the viewing duration as in [41] was considered as an indicator for the signal strength and methods were proposed to deal with the uncertainty of the signal. The case of *linear* programs in contrast to video-on-demand services was, e.g., discussed in [133] where they also consider various information signals related to noise in the data and the new-item problem. In the deployed TiVo system [4], the fact that someone recorded a show is treated as an implicit feedback signal and combined with explicit binary feedback. According to the recent literature review in [120], implicit profiling is therefore the most common approach in this domain.

[2] `http://recsys.acm.org/recsys15/challenge/`

Music Recommendation. The use of implicit feedback signals for music recommendation and playlist generation will be discussed in more depth in Section 14.6.2. As an example, consider the work presented in [87] where the authors develop a multi-criteria music recommendation approach which utilizes both explicit as well as implicit feedback. Implicit feedback signals are inferred both for the overall rating of the track as well as for the criteria preferences (i.e., on music, lyrics and voice). As feedback signals the authors use the total time spent by users hearing a track, the number of accesses to an item and the actual play duration per listening event.

Another music-related approach is presented in [64], where the authors as in [18] rely on listening logs of users obtained from the Last.fm music platform as a basis for music recommendation. A specific aspect of their work is that their algorithms exploit additional (time-related) context information which they automatically derive from logs.

14.3.1.4 Social Behavior. With the development of the "participatory Web", social networks, and Web 2.0 technologies, users transformed from being pure information consumers to becoming also active content contributors. They now can explore the information space of the Web not only by accessing the structures provided by (classic) information providers, but also by using the behavior or content from other peers in their social networks as guidance. Typical interactions of this "social navigation" are, for example, commenting or posting on a social network or microblogging platform, tagging or bookmarking content on the Web or establishing social connections with other people [40].

Given these novel types of interactions, a number of additional preference signals can be used in recommendation processes. Some of these types of signals were anticipated in the *Annotation* and *Create* categories of observable behavior in [84] and [52]. Since the observable user actions on the Social Web are not necessarily directly related to a target object (such as "annotate" or "publish") but can signify also indirect preference indications, we introduce "*Social & Public Action*" as an additional category.

Tags and Bookmarks. Bookmarking or tagging items with keywords for own later use is a classic implicit feedback signal in Information Filtering. In the Social Web sphere, tags and bookmarks are now shared with others and can serve as a basis, e.g., to build tag-based recommender systems [33, 131, 110, 28], see also Chapter 12 of this book [8].

Posts and Comments. Publishing information on social media in terms of a post or comment about an opinion or the own current activity is another type of implicit preference signal on the Social Web, as further discussed in Chapter 10 of this book [55]. Such often very short posts can be analyzed to build user profiles that reflect the user's interests [1]. The contents of posts was for example analyzed in [96] through a topic modeling technique with the goal to recommend other users to follow on the social network (see also Chapter 15 of this book [31]). Finally, the problem of filtering interesting items in a social "feed" corresponds

to a classic collaborative information filtering problem with some additional challenges, e.g., that the content to be analyzed can be very short [117].

Structuring Objects. The organization of objects for later use is another observable user action mentioned in [84]. A typical example in the recommendation domain is when users share music playlists, which can serve as a basis for next-track music recommendation [11].

Connecting with Others. A final category of implicit feedback signals can be the user's embedding within a social network. One can analyze the user's social neighborhood, explicit or implicit trust signals, or the network topology [5] to recommend additional friends or followees, or inspect existing group memberships or channel subscriptions and their topics to recommend further groups or other items [32][3], as discussed in more detail in Chapter 15 of this book [31]. In [71], for instance, the followers of Twitter accounts are used to generate interest profiles in the context of the cold-start problem for app recommendation. Another application domain, personalized social search, exploits an individual's relations in a social network to compute more relevant query results [14].

14.3.1.5 Ubiquitous User Modeling. With the availability of modern smartphone devices and their various sensors as well as the emerging trend of the "Internet of Things", more and more information about the user's current location and environment becomes available. We propose to summarize these types of observable user actions under the umbrella term "*Physical Action*" in the extended classification scheme.

Location and Movement Profiles. The user's past and current movement profile can be a valuable indicator of the user's interests, as also discussed in Chapter 16 of this book [12] in the context of social media data. In [9], for example, the movement profiles and dwelling times of users in a museum are used as indicators for the user's interest in the individual exhibit objects. Other application domains in which the past locations of the users can be used for user profiling include in particular the tourism domain – think, e.g., of past visited places or GPS trajectories [132] as interest indicators – or leisure activities. In the mobile context a proactive approach has been advocated to enrich and ease the mobile experience by "*providing the right information, at the right time, and in the right form for the current context*" [115]. However, such a proactive system behavior exclusively relies on implicit user feedback and accurate observations of user actions in order to avoid an obtrusive system behavior [29]. The work in [67] analyzed, for instance, a user's activity (movement) from GPS logs in order to develop a proactivity model and determine when it is appropriate to interrupt the user and to provide an unrequested recommendation.

Note that in contrast to context-aware recommendations (CARS) we are not necessarily interested in the user's *current* location to make suitable recommen-

[3] As indicated in Section 14.2, we consider such information only as implicit feedback if the signal is related to some target recommendation object.

dations, but rather rely on the observed user behavior and relationships between past user actions to determine the appropriate next steps.

Smart Homes. In the Internet of Things, all sorts of electronic devices, e.g., in a smart home, will be connected with the network and can represent additional sources of information about the environment of a user or with which devices the user has interacted with. One typical task in such a context is called "activity recognition", i.e., to estimate based on the available sensor data, e.g., from a mobile phone [23, 95], which activity the user currently pursues and where he or she is located. Quite an amount of research on knowledge-based or learning-based activity recognition has been done in the area of *smart homes*, see e.g. [118] for an early work. While these types of sensor information have been largely ignored in the mainstream recommender systems literature, these preference-based adaptations of, for instance, light or music actors [53] in smart homes represent adaptive and personalized systems in their purest form that heavily rely on implicit user feedback. Some more recent examples include the automatic identification of users while watching TV in order to learn their interests [72] or the use of gaze tracking in combination with explicit ratings to derive content-based interest profiles [54].

14.3.2 An Extended Categorization of Observable User Behavior

Oard and Kim's early categorization scheme – *Examination*, *Retention*, *Reference* – was mainly focusing on document-centric applications and is in particular suitable when the goal is to recommend news messages, text documents, or web pages, see Table 14.2. This also holds for the additions in [84] and [49], *Annotation* and *Create*. With the wide-spread application of recommendation technology in all sorts of domains that we have observed in the last two decades, a variety of other types of implicit feedback signals have been successfully exploited since Oard and Kim's early work.

Based on our review of application scenarios for implicit feedback in recommender systems, we suggest to extend the existing classification scheme with additional observable user actions. They are related to (a) the user's social behavior and (b) the increased availability of data for "ubiquitous" user modeling. The new items are shown in Table 14.2 with a detailed description. They meet the requirements of new behavioral patterns that emerged with the widespread availability of connected mobile devices and social functionalities on the Web. Keep in mind that in practice the types of observable behavior can overlap. For example, the *Social & Public Actions* "posting" and "rating" articles on a social network can also be seen as *Create* and *Annotation* actions.

Table 14.2 Extension of the five types of observable behavior (see Table 14.1 and [84, 85, 52]) by two now categories: *Social & Public Action* and *Physical Action*.

Category	**Examples of Observable Behavior**
Examination	Duration of viewing time, repeated consumption, selection of text parts, dwell time at specific locations in a document, purchase or subscription.
Retention	Preparation for future use by bookmarking or saving a named or annotated reference, printing, deleting.
Reference	Establishing a link between objects. Forwarding a document and replying, creating hyperlinks between documents, referencing documents.
Annotation	Mark up, rate or publish an object (includes explicit feedback).
Create	Write or edit a document, e.g. [39] or [13].
Social & Public Action	Public posting, commenting and communicating, activity posts, following and connecting with people, joining groups, expressing trust.
Physical Action	Observed user actions that can be interpreted as feedback towards objects of the physical world. Being at a location, roaming profiles and dwelling time, other recognizable activities in the physical world (e.g., smart homes, Internet of Things).

14.4 Algorithms for Implicit Feedback Situations

In this section, we will discuss algorithmic approaches to generate recommendations based on implicit feedback. As mentioned earlier, interpreting implicit feedback can be difficult and we will first discuss techniques to transform and encode preference signals as explicit feedback to be able to use standardized recommender system algorithms for rating prediction. After that, we will briefly present selected examples of collaborative filtering algorithms that are especially designed to deal with "one-class" only feedback signals. Then, we will examine methods to find frequent patters in implicit feedback in more detail and finally show examples of hybrid algorithms that try to combine explicit ratings with implicit feedback.

14.4.1 Converting Implicit Signals to Ratings

As discussed in Section 14.2.3, implicit feedback is often "positive-only", i.e., no or only minimal information is given about items that were disliked by the users. Also, there are often multiple signals and different kinds of feedback, e.g., when a user visits an item detail page in an online store multiple times, bought some items and placed other items to a "wishlist". In addition, the "rating matrix" of implicit feedback is most of the time very sparse. Therefore, the available data consists of few positive signals that are sometimes hard to interpret and numerous unlabeled examples.

To deal with such situations, many so-called "One-Class Collaborative Filtering" techniques were proposed in the literature, some of which we discuss later in

Section 14.4.2. In this section, we show an alternative to this approach, which is the transformation of the given data into two-class or multilevel numerical "ratings" and the creation of a user-item rating matrix. Such a transformation then allows us to apply standard recommendation techniques which were originally designed for explicit ratings. In the following, we will discuss ways of transforming implicit feedback signals into numerical rating values. Further discussion on rating-based collaborative filtering approaches can be found in Chapter 10 of this book [55].

14.4.1.1 Problems of Basic Transformation Strategies. The first step of a basic implicit-to-numerical transformation is to add a virtual rating of "1" to the user-item rating matrix for each observed user-item interaction. Different options exist to deal with the unknown data points and the missing negative feedback, each of them having certain drawbacks as discussed in [88].

All Missing as Negative (AMAN). In this approach, all non-observed items are treated as a "0" rating and thus as negative feedback. The resulting user-item-matrix contains only ones and zeros, see Figure 14.2. When a machine learning model is fitted to this data, the distribution between the two classes – 0 and 1 – is strongly biased toward the negative feedback, since there are only few positive entries in the matrix. Any rating prediction technique for explicit feedback might tend to always predict 0. Usually, regularization methods are used to prevent this kind of overfitting but the ratio of negative to positive feedback in the data is however still problematic [101].

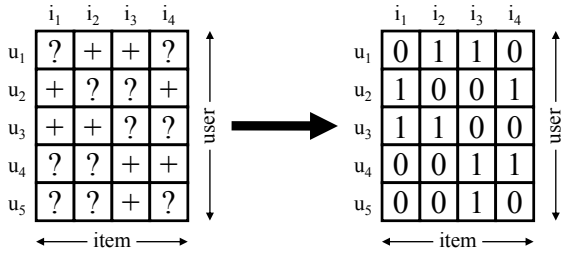

Fig. 14.2 AMAN: Transformation of implicit feedback to explicit "0" and "1" ratings [88, 101].

All Missing as Unknown (AMAU). Alternatively, the missing data points could be treated as unknowns, see Figure 14.3. A rating prediction algorithm therefore only operates on the positive ratings, i.e., ignores all missing data. Since the whole dataset only consists of "1" ratings, the distribution is biased toward the positive feedback. Without proper regularization, this would result in a trivial solution and typical explicit feedback algorithms would tend to always predict 1 [116].

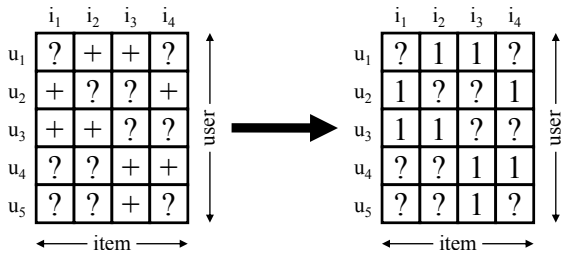

Fig. 14.3 AMAU: Transformation of implicit feedback to explicit unknown and "1" ratings [88, 101].

14.4.1.2 Discerning Negative from Unknown Signals. To avoid the drawbacks of the *extreme* ways to deal with unknown examples – labeling them as negative or ignoring them – more advanced approaches assume that there might be *some* negative examples in the unknown data. If these could be labeled properly, existing explicit feedback approaches could be employed.

Several ways to guess which of the unknown entries could be negative feedback have been introduced in the past, some of them as part of a one-class collaborative filtering techniques which we will discuss in Section 14.4.2.

A simple approach is to randomly sample negative examples from the unknowns, as done, e.g., in [101]. To learn the ranking of items, their approach uses positive-negative item-pairs for each user. However, since there is no negative feedback in the data, they select a random unknown item for each (positive) feedback to create the pairs. More elaborate schemes use statistical [92] or weighting-based [88] approaches to choose negative samples in a way that the distribution of the resulting set of negative ratings resembles the set of the positive ratings.

As an alternative to inferring negative ratings, users could be asked to give some negative (and positive) feedback, e.g., in an initial interaction phase with the system. However, this would be considered explicit feedback and might be perceived as a burden by users [88, 94].

14.4.1.3 Converting Graded Implicit Feedback to Ratings. Instead of converting implicit feedback signals into explicit 0/1 ratings, some proposals in the literature adopt more fine-grained strategies. Since in many application settings different types of user behavior can be observed, the idea is to assign a different "strength" to each type of signal, i.e., to encode the different levels of *graded* relevance feedback as ratings.

In a study on recommendations in an online mobile games store [42], for example, the authors used explicit ratings – which were only sparsely available – and in addition considered view and purchase actions, which were transformed into explicit rating values. On a scale from -2 to $+2$, view events were interpreted as 0 and purchase events as $+1$. Explicit positive (negative) ratings where considered as a $+2$ (-2) rating. The choice of this encoding was done somewhat

arbitrarily and led to a very skewed distribution of the rating values, as there are many more view events than purchase events.

A time-based approach of assigning numerical values to implicit feedback signals was proposed by Lee et al. [65] in the context of a recommender system for wallpaper images. Purchase information was used as an implicit signal and the strength of the signal was determined based on the release date of an item and the point in time when the user made the purchase. The authors then used a time-based decay function to promote more recent events received higher scores.

An approach of combining different feedback types was presented by Parra and Amatriain [93] in the context of music recommendation. The authors propose to use a linear regression model to combine three different aspects of implicit feedback signals – personal feedback, global feedback and recentness – into a rating score. They conclude that the former two interaction types have the strongest impact on the recommendation accuracy. In [94], this model is extended to a logistic regression model which includes a number of additional variables related to consumption behavior as well as demographic data.

In [59], another work in the field of music recommendation, items that have both explicit ratings and observed user actions are exploited to learn which types of implicit feedback can be mapped onto which ratings. The user actions that were interpreted as implicit feedback consist of play counts and play percentages, listening date and time, number of skips and next-track statistics. Subsequently, the system rates items that did not receive any explicit feedback with a naive Bayesian classification based on the implicit signals that the item received.

The examples above show that transforming signals of different types of user behavior into one single (rating) score largely depends on the respective domain and cannot be generalized easily. Sometimes, it may not be possible to map different kinds of feedback, e.g., viewing and buying an item, to a linear rating scale. Other signals may be difficult to interpret, for example, a short dwelling time for an item detail page could be interpreted as negative feedback because the user seems to be not interested in the item. However, it could also mean that the user already knows the item and does not need to look at the page again without an indication of positive or negative feedback. On the other hand, a long dwelling time does not necessarily correspond to positive feedback. A user might have lost interest and abandoned the page without actively leaving it because the item was not relevant anymore. In the following section, we therefore discuss the correlation between implicit signals and explicit ratings in more detail.

14.4.1.4 Correlating Implicit Feedback with Explicit Ratings. How to encode different types of feedback into numerical scores can, as discussed, be challenging and is sometimes done in an arbitrary manner. Several researchers have therefore investigated the relationship between explicit ratings and implicit feedback actions, including [19, 93, 94, 99, 112]. Depending on the domain and experimental setup, the obtained results are however not always consistent.

In [19], the results of a laboratory study are reported in which users were first asked to freely browse the Web for 30 minutes and subsequently had to rate each visited page with respect to how interesting its contents were. The recorded

user actions, such as mouse movements and dwelling times, were then compared with the collected explicit ratings. The analysis revealed that the time spent and the scrolling activity on a web page correlates with explicit ratings. Other indicators, however, such as mouse movement and clicks, had no clear relation to the participant's interest.

Zhang and Callan [129] report the results of a user study on a web-based news filtering system. The participants had to read personalized news for one hour per day over a period of 4 weeks and assess the articles according to multiple dimensions, such as relevance, novelty and readability. After the study, each participant also completed a questionnaire about the topics of the articles that they read. In addition, the same user actions as in [19] were recorded and the authors similarly concluded that dwelling time and scrolling activity correlate the most with the explicit ratings. However, they also state that the answers of the questionnaire about topic interests are much more correlated to the explicit feedback and therefore advise that in real-world settings the users should initially be asked about their topics of interest.

Building on the insights and log data from this study and the work from [19], the authors of [134] propose a Bayesian modeling technique to combine implicit and explicit feedback signals. Their results, however, indicate that the implicit feedback was unstable and possessed only limited predictive value. Thus, the combination of both feedback types was only marginally better than when using explicit feedback alone.

The results of a similar study on the relationship between various types of browsing actions and explicit interest statements are reported in [112]. The strongest correlations with the explicit ratings were found for the indicator "time of mouse movement relative to reading time" and the number of visited links on the page. Note that mouse movements were not considered to be a good indicator according to the study [19] discussed above that however did not put the mouse movements in relation to the dwelling time.

More recently, Parra et al. [93, 94] report on their attempt to correlate implicit and explicit preferences in the music domain. The authors first carried out a survey in which the participants rated tracks from Last.fm. This information was used to derive preference patterns and biases, e.g., whether users generally prefer recent or popular tracks. The insights of the survey were then used to design a linear regression model to predict ratings from *what they call* implicit feedback signals. In fact, the authors rather adopt an approach based on metadata to learn which features of the liked items are particularly relevant to the users.

Finally, in some domains implicit feedback seems to be more meaningful than explicit preference information. In [99], Pizzato et al. use the data of 21.000 users of an online dating platform and compare the predictive accuracy of different input types. In contrast to most of the other works reviewed so far, their results show that explicit preference statements are often incomplete or imprecise and recommending based on implicit feedback can be more accurate. This emphasizes

once more that the interpretation of implicit feedback can be highly dependent on the respective domain.

14.4.2 One-class Collaborative Filtering Techniques

The naive conversion strategies to generate (binary) numerical scores from implicit feedback have their drawbacks, e.g., converting all unobserved data points into zeros (*All Missing as Negative*, AMAN) or leaving them as unknowns (*All Missing as Unknown*, AMAU) both result in a class imbalance problems and standard rating prediction techniques tend to always predict 0 or 1, respectively. Therefore, more sophisticated techniques were proposed to deal with positive-only feedback in the literature.

These so-called *one-class collaborative filtering techniques (OCCF)* [88] are algorithms that only need one single type of signals. They usually interpolate which of the missing data points could be negative feedback or try to guess if a user prefers one item over a different, unknown one. Typically, OCCF techniques are used in domains where only unary implicit feedback is available. As discussed earlier in Section 14.2.1, some of them are also applicable when dealing with unary explicit feedback, such as "likes" on a social network, as well. In the following, we will present examples of selected OCCF techniques in more detail – wALS, Random Graphs, BPR, CLiMF – and briefly review other related approaches.

14.4.2.1 wALS and sALS-ENS. In [88], the authors introduce two strategies to handle the missing feedback in a way that is somewhere in between the two extremes of AMAU and AMAN. In the first strategy, a low-rank approximation $\mathbf{X}$ of the "rating" matrix R is calculated and in the objective function, a confidence weight is used to express the probability that a signal is (correctly) interpreted as positive or negative. A weight of 1 is assigned to the positive data points, since they are known beforehand. The unknown, missing values, on the other hand, have a confidence value lower than 1, because some of them have the chance to be negative samples. The following equation shows the objective function.

$$\mathcal{L}(\mathbf{U},\mathbf{V}) = \sum_{ij} W_{ij}(R_{ij} - \mathbf{U}_i\mathbf{V}_j^T)^2 + \lambda(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2) \tag{14.1}$$

The low-rank approximation $\mathbf{X}$ of R is decomposed to $\mathbf{X} = \mathbf{U}\mathbf{V}^T$ and can be used for prediction. To prevent overfitting, the objective function has a regularization term. The matrix W is the non-negative weight matrix that assigns confidence values to the observations and the optimization problem is solved by an Alternating Least Squares algorithm (ALS), hence the name wALS (weighted ALS).

The characteristics of this algorithm are influenced by the the choice of W. For $W = 1$, the confidence for all data points would be 1 and therefore the strategy would be equivalent to AMAN, where all unknown are treated as negatives. The authors propose three different weighting schemes W for the unknown entries W_{ij} in the user-item interaction matrix which are described in Table 14.3.

Table 14.3 Weighting schemes for OCCF.

Weighting scheme	**Confidence matrix**	**Description**
Uniform	$W_{ij} = \delta$	All missing entries are assigned a fixed confidence weight δ between 0 and 1.
User-Oriented	$W_{ij} \propto \sum_j R_{ij}$;	Higher confidence is set for "heavy" users as they are assumed to know the item catalog better and therefore have discarded unknown items with a higher probability.
Item-Oriented	$W_{ij} \propto m - \sum_i R_{ij}$;	Higher confidence is set for items which received more interactions, i.e., unpopular items have a higher probability to be discarded as negative. Here, m is the number of users.

Calculating a large approximative low-rank matrix is however computationally intensive and, in addition, the class imbalance problem is still present, because there are still many more negative than positive samples.

Therefore, with sALS-ENS the authors propose a more advanced way to consider all (known) positive examples from the data and add a subsample of negative feedback based on a sampling probability matrix. They propose three sampling strategies that behave similar to the ones used for the weighting matrix of wALS. As a result, a smaller rating matrix is generated that can be used as a basis for calculating the low-rank approximation of R via ALS. The experiments show that the wALS approach is slightly superior in terms of accuracy but considerably slower than sALS-ENS. An approach similar to wALS has been proposed in [41] that also uses a weighting term for the implicit observations.

14.4.2.2 Random Graphs. In a similar spirit, Paquet and Koenigstein [92] model the unknown negative feedback using a random graph. The approach is based on a bipartite graph G that contains edges $g_{mn} = 1$ between users m and items n when observed implicit feedback is available.

The additional assumption is however that, although a user m has interacted with an item n, there should be some other items that the user considered but discarded as not relevant. The authors therefore model a second *hidden* graph H that is also bipartite and contains edges $h_{mn} = 1$ whenever a user m considered an item n. In addition, $g_{mn} = 1 \Rightarrow h_{mn} = 1$ holds, i.e., if a user m accepted an item n, then it was considered before.

Therefore, G is a subgraph of H and all the other edges of H are the considered items that were discarded as not relevant, i.e., the negative feedback. Since the negative feedback is unobserved, the authors use the following popularity-

based sampling strategy to generate the edges in H that represent the negative feedback.

For each user m with d_m observations of positive feedback, additional d_m edges of negative feedback are randomly sampled from a distribution $M(\pi)$ based on the popularity of all items. Instead of using a popularity distribution with $\pi_n = d_n$, where d_n is the number of times that there was positive feedback for an item n, the authors assume that popular items are generally more liked, i.e., have less negative feedback. Therefore, for the popular items, less negative examples should be sampled for H and the distribution is modified in the following way: $\pi_n = d_n^\gamma$ with $\gamma = 1 - \log d_{max} / \log r$, where $d_{max} = max\{d_n\}$.

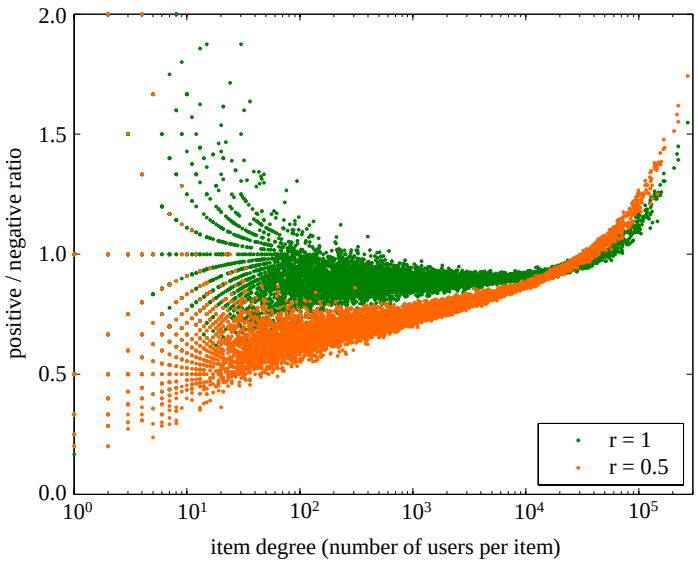

Fig. 14.4 Ratio of positive and negative edges in H [92].

The parameter r controls the ratio between sampled negative and known positive samples in H, as can be seen in Figure 14.4. In addition, the sampling procedure is done "without replacement", i.e., if the sampling draws an already known positive example $h_{mn} = 1 \wedge g_{mn} = 1$, no negative sample $h_{mn} = 0$ is added. Therefore, the ratio is skewed for the most popular items, as there is a higher chance to draw a positive example for them, which results in a lower amount of generated negative feedback signals.

An advantage of this approach is that it is easily extensible with richer feedback signals. For example, the hidden graph H could also be populated with implicit negative examples gathered from other sources, e.g., a visited detail pages of an item without a subsequent purchase or a purchase of an equivalent item could indicate $h_{mn} = 1 \wedge g_{mn} = 0$. Similarly, information about items that could have never been considered $h_{mn} = 0$ could be included in the graph, e.g., because the item was not listed in the shop at the time the user was active.

To generate recommendations with the two graphs G and H the authors propose a bilinear collaborative filtering model with matrix factorization which can estimate the probability of accepting an item as relevant after considering it under $p(g_{mn} = 1|h_{mn} = 1)$. The model is designed to be largely agnostic of the popularity of the items. Therefore, the popularity bias of the recommendations can be reduced. More details on the model have been discussed in [92]. The proposed approach was developed and deployed in the context of the Microsoft Xbox Live environment, in which one particular challenge lies in the large amounts of data that have to be processed.

14.4.2.3 Bayesian Personalized Ranking (BPR). BPR [101] – as already mentioned in Chapter 10 of this book [55] – deals with the one-class CF problem by turning it into a ranking task and implicitly assuming that users prefer items they have interacted with over other unknown items.

In some sense, BPR therefore creates artificial negative feedback in a similar spirit as the approaches discussed so far. However, instead of applying rating-prediction techniques using the implicit feedback data BPR ranks the candidate items for a user without calculating a "virtual" rating.

The overall goal of the algorithm is to find a personalized total ranking $>_u \subset I^2$ for all users $u \in U$ and pairs of items $(i, j) \in I^2$ that has to satisfy the properties of a total order (totality, antisymmetry, transitivity).

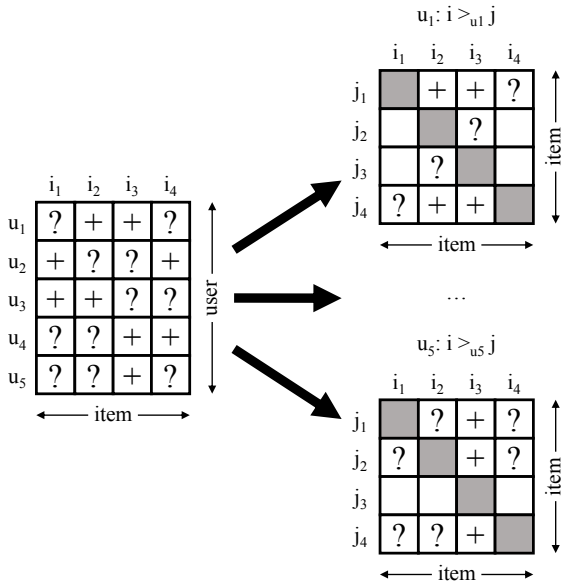

Fig. 14.5 Transformation of implicit feedback to pairwise preferences for each user [101].

To model the negative feedback, Rendle et al. [101] use a pair-wise interpretation of the positive-only feedback. The general idea is that a user's positive

feedback for an item is interpreted as the user's preference of this item over all other items that the user did not give feedback for. As shown in Figure 14.5, the positive-only feedback is thus transformed to positive and negative feedback in pairs of items (i, j) where the user preferred i over j (positive), or j over i (negative). If the user interacted with both items or none of them, no additional information can be deduced for the pair (i, j). The different pairs of items form the training data D_S for the BPR algorithm and can be formalized as triples of a user and an item pair:

$$\begin{aligned} D_S &:= \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\} \\ &\text{with} \\ I_u^+ &: \text{items with implicit feedback from } u \end{aligned} \tag{14.2}$$

To create a personalized ranking of items, the authors introduce a general optimization criterion called BPR-Opt, which is derived through a Bayesian analysis of the problem and which aims to maximize the posterior probability $p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta)$ where Θ is the parameter vector of the underlying algorithmic model. The optimization criterion, including substitutions for smoothing, is formulated as follows:

$$\begin{aligned} \text{BPR-Opt} &:= \ln p(\Theta| >_u) \\ &= \ln p(>_u |\Theta)p(\Theta) \\ &= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \\ &\text{with} \\ \sigma(x) &= 1/(1 + e^{-x}) \\ \hat{x}_{uij} &:= \hat{x}_{ui} - \hat{x}_{uj} : \text{a model-specific relationship function} \\ \lambda_\Theta &: \text{model specific parameters} \end{aligned} \tag{14.3}$$

The BPR-Opt criterion is related to the AUC metric and optimizes it indirectly. To solve the optimization problem, a gradient descent on the model parameters Θ can be used. Since it is computationally expensive to take all triples $(u, i, j) \in D_S$ into account, a stochastic gradient descent approach randomly chooses the triples uniformly from D_S.

By decomposing the model specific function $\hat{x}_{uij}$ – which is a real-valued function for the relationship of the items i and j for user u – into $\hat{x}_{ui}$ and $\hat{x}_{uj}$, existing techniques for rating prediction can be applied to calculate the two terms. In [101], both a matrix factorization model and a kNN approach are presented as the underlying model for the BPR algorithm. Compared to standalone Matrix Factorization (MF) or kNN models, which minimize the rating prediction error, the BPR-Opt criterion instead ensures that the item ranking is optimized.

14.4.2.4 CLiMF. *Collaborative Less is More Filtering* is another approach for ranking optimization in one-class CF settings [114]. CLiMF aims to directly optimize a smoothed version of the Mean Reciprocal Rank (MRR) metric to achieve an optimal ranking of the top-n items. In comparison to BPR, both algorithm optimize a smoothed version of a ranking metric. BPR, however, implicitly assumes negative feedback in the data, while CLiMF only uses the positive feedback signals.

The optimization target of CLiMF, the reciprocal rank RR_i of a recommendation list for a user i, represents the position of the earliest occurrence of a relevant item for the user. For N items it can be defined as:

$$\begin{aligned} RR_i &= \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij})) \\ &\text{with} \\ Y_{ij} &= 1 \text{ if } i \text{ interacted with } j, \text{ else } 0 \\ R_{ij} &= \text{rank of item } j \text{ in list of user } i \\ \mathbb{I}(x) &= 1 \text{ if } x = true, \text{ else } 0 \end{aligned} \tag{14.4}$$

In essence, the formula only calculates $1/R_{ij}$ for the first relevant item j for user i. However, directly optimizing the reciprocal rank with standard optimization functions – like gradient descent – is not possible, since it is a non-smooth function. Therefore, the authors introduce a smoothed approximation of RR_i which can be optimized. To that end, the indicator function $\mathbb{I}(x)$ and the rank $1/R_{ij}$ are substituted by the following approximations:

$$\begin{aligned} \mathbb{I}(R_{ik} < R_{ij}) &\approx g(f_{ik} - f_{ij}) \\ 1/R_{ij} &\approx g(f_{ij}) \\ &\text{with} \\ g(x) &= 1/(1 + e^{-x}) \\ f_{ij} &= \langle U_i, V_j \rangle \end{aligned} \tag{14.5}$$

Here, the predictor function for the relevance score f_{ij} is based on a factor model of the latent user and item factor vectors U_i and V_j. Although inserting the substitutions of Equation 14.5 in Equation 14.4 creates a smooth approximation of the reciprocal rank and could in theory be optimized, the optimization task has a complexity of $O(N^2)$, i.e., is quadratic with the number of items, which is not practically feasible in most domains. It is, however, possible to derive a lower bound of the reciprocal rank which can be optimized with a lower complexity [114]:

$$L(U_i, V) = \sum_{j=1}^{N} Y_{ij} [\ln g(f_{ij}) + \sum_{k=1}^{N} \ln(1 - Y_{ik} g(f_{ik} - f_{ij}))] \tag{14.6}$$

The optimization function of CLiMF (Equation 14.6) has two terms that are maximized: (1) Y_{ij} and (2) the rest of the equation in square brackets. Maximizing the first term promotes the relevant items. Maximizing the second term optimizes the ranking by learning latent factors. As discussed in [121], this can also lead to a diversification of the recommendation results. Equation 14.7 shows the final regularized optimization function for all users. It can be optimized with a stochastic gradient descent approach and a complexity of $O(dS)$ with S being the number of observed positive feedback examples.

$$F(U,V) = \sum_{i=1}^{M}\sum_{j=1}^{N} Y_{ij}[\ln g(U_i^T V_j) + \sum_{k=1}^{N} \ln(1 - Y_{ik} g(U_i^T V_k - U_i^T V_j))] \\ - \frac{\lambda}{2}(||U||^2 + ||V||^2) \tag{14.7}$$

Later on, the authors proposed a generalized version of CLiMF called xCLiMF which is able to deal with situations where a relevance level for the feedback is available [113]. A similar generalization to deal with graded relevance feedback was also proposed for BPR in [66], which will be discussed later in Section 14.6.3.2.

14.4.2.5 Other One-Class Classification Approaches. The problem of positive-only data can also be found in the field of classification when we only have positively labeled training data. Support Vector Machines (SVM) are a typical method that was originally designed for two-class classification tasks and requires labeled input data. In [109], Schölkopf et al. develop a theoretical foundation to apply support vector machines (SVM) to unlabeled, one-class data.

These one-class SVM (1-SVM) are able to identify a region in the item-space where most of the "positive" examples are located. Likewise, the other regions of the item-space can be labeled as "negative". A practical implementation of 1-SVM for recommender systems and a benchmark on the MovieLens data is presented in [125]. Similar examples for classification tasks without the need for negative training samples are [126] and [51], where the goals are to classify websites and text based on positive and unlabeled data only.

A density estimation that is similar to the one of Schölkopf et al. [109] is presented in [7]. The authors introduce a model to estimate high-density areas of the data points in the item-space. Additionally, the model assumes that not all the data points are positive feedback but could also be negative examples. If there is positive feedback as well as unlabeled examples in the data, it is possible to solve the one-class classification problem by applying the expectation-maximization (EM) algorithm, see, e.g., [123] and [24]. In [69], finally, unlabeled data points are treated as negative examples. This transforms the problem into a problem of "learning with noise" and is solved with a regression approach to model a linear function as a classifier.

14.4.3 Frequent Patterns in Implicit Feedback

One of the most prominent examples of a recommendation system is Amazon.com's list of shopping proposals labeled "Customers who bought ... also bought ...". The label suggests that the contents of the non-personalized but item-dependent list are based on the analysis of the buying behavior of Amazon's customer base and the detection of item co-occurrence patterns.

In these classic *Shopping Basket Analysis* settings, the goal is to find sets of items that are frequently purchased together. The input to the analysis is a set of purchase *transactions* where each transaction contains a set of items that were bought together, e.g., in one shopping session.

14.4.3.1 Association Rule Mining. Technically, the identification of such patterns can be accomplished with the help of *Association Rule Mining* (ARM) techniques [2]. An association rule has the form $A \Rightarrow B$, where A and B are sets of items and the arrow can express something like "whenever A was purchased, also B was purchased"; typically, the strength of a rule is expressed in terms of the measures *support* and *confidence*.

Following the description of [108], *Association Rule Mining* can be formally defined as follows. Let $T = \{t_1, \ldots, t_m\}$ be the set of all transactions and $I = \{i_1, \ldots, i_n\}$ the set of all available items. Each transaction t consists of a subset of items $t \subseteq I$. A transaction t could therefore represent a shopping basket of items that was bought by a customer. Let $A, B \subseteq I$ and $A \cap B = \emptyset$, i.e., A and B are also subsets of I but have no items in common. An *association rule* is defined as the implication $A \Rightarrow B$. It expresses that whenever the items contained in A are included in the transaction t, then items contained in B will also be included in t. The the left side of a rule $A \Rightarrow B$ is often called the *rule body* or *antecedent* while the right side is the *rule head* or *consequent*.

As can be seen from the definition above, each co-occurrence of two or more items in a transaction can be expressed as an association rule. However, not all association rules are helpful, e.g., two items could only have occurred together once in a single transaction, and the goal of *Association Rule Mining* is to find only those rules in a set of transactions T that are meaningful. To quantify the significance of association rules, various measures have been introduced in the past[4] but the most widely-used measures are *support* and *confidence*.

The *support* of a set of items A is the proportion of transactions $t \in T$ that contains A, i.e., the transactions where $A \subseteq t$. It can also interpreted as the probability of the co-occurrence of all items in A in a transaction.

$$supp(A) = \frac{|\{t \in T; A \subseteq t\}|}{|T|} \tag{14.8}$$

The *confidence* of an association rule is then defined as the ratio between the number of transactions that contain $A \cup B$ in relation to the number of transac-

[4] `http://michael.hahsler.net/research/association_rules/measures.html`

tions that only contain A. Therefore, the *confidence* is the conditional probability of B given A. It can therefore be expressed with the support of A and $A \cup B$.

$$conf(A \Rightarrow B) = \frac{supp(A \cup B)}{supp(A)} \tag{14.9}$$

Usually a minimum support and a minimum confidence has to be satisfied by an association rule to be considered significantly meaningful for the transactions. Therefore, when generating the association rules, first a threshold for the support is applied to find the most frequent sets of items in the transactions. However, when there are n items in the set of all items I, the number of possible subsets that have to be considered is $2^n - 1$, excluding the empty set. For a large number of items n, considering all combinations individually is not feasible. Efficient calculation of the support is however possible by exploiting the *downward-closure property* [2]: If an itemset A is frequent according to some support threshold, then all of its subsets $A' \subseteq A$ are also frequent for that threshold. Likewise, if an itemset A is not frequent according to some support threshold, then all of its supersets $A' \supseteq A$ are also not frequent for that threshold. After the most frequent itemsets have been found, a confidence threshold is used to determine the most important association rules.

To detect these rules automatically and for large amounts of data, a variety of algorithms was proposed over the last decades to find frequent patterns and derive association rules, starting with the Apriori algorithm [2] that uses the *downward-closure property*, over more efficient schemes like FP-Growth [34] to techniques that find patterns in parallel [68] or are able to identify rules for niche items [73]. For cases in which the sequence of the item interactions is relevant, *Sequential Pattern Mining* [3] can be applied.

14.4.3.2 Recommending with Association Rules. Once the rules are determined, recommending based on association rules can be done, e.g., in the e-commerce domain, as follows. First, we determine the set of the current user's (recently) purchased or viewed items and then look for rules in which these items appear in the antecedent (the left hand side, A). The elements appearing in the right hand sides (B) of the corresponding rules then form the set of possible recommendations. Items for recommendation can then be ranked with a $score_{ui}$ based on different heuristics, e.g., by using the confidence of the rules that are applicable to the subsets A of the items I_u that a user u purchased and that lead to the inclusion of an individual item i.

$$score_{ui} = \sum_{A \subseteq I_u} conf(A \Rightarrow i) \tag{14.10}$$

A specific aspect to consider in the recommendation domain is that we are not necessarily interested in the strongest rules, as they might lead to obvious recommendations, but rather in rules for unexpected patterns or niche items. Also, depending on the domain, different kinds of association rules can be mined. In [73], for example, the score used to rank the items was calculated using both

user associations ("user u_1 likes an item" ⇒ "user u_2 likes an item") and items associations ("item i_1 is liked" ⇒ "item i_2 is liked"). When recommending, their approach ranks the items by a $score_{ui}^{User}$ for users that are above a fixed support threshold, i.e., users that already have left some feedback in the system. The score is calculated both from the support and confidence of the user associations. However, A is now a subset of the users U_i that liked the item i.

$$score_{ui}^{User} = \sum_{A \subseteq U_i} supp(A \cup u) \cdot conf(A \Rightarrow u) \tag{14.11}$$

For users below the support threshold, i.e., cold-start users that only left little feedback, item association rules are used to calculate the item ranking similar to Equation 14.10. To find niche items, the item association rules are however mined for each item as a fixed *consequent* and only the subset of transactions T' that contains the *consequent* is used to calculate the support of each item. Since T' is usually small compared to T, the resulting support for the items is higher. Otherwise, rules for new or niche items would be filtered out, since their support would often be below the general support threshold over all transactions T.

Amazon.com's "Customers who bought" recommendations can be generated in a similar way with Association Rule Mining. The particularity of such an approach is that only frequent itemsets of size two are required – which means that simple co-occurrence counts can be sufficient – and that these recommendations can already be provided in the context of the customer view of one particular item.

Association Rule Mining techniques have been applied in different recommendation scenarios in the literature. Examples include the identification of navigation patterns in the context of Web Usage Mining [80], the identification of rules exploiting item characteristics in e-commerce [98], the recommendation of next tracks in music playlist generation [11, 36], or in the context of e-learning [27]. Association Rules and "co-visitation counts" also serve as a basis for the YouTube video recommendation system [21].

14.4.4 Hybrid Implicit-Explicit Techniques

In some domains both explicit and implicit feedback signals are available. For example, in most online stores users can rate products and at the same time their navigation behavior is logged by the system. In the following sections, we will discuss some approaches that combine explicit and implicit feedback or use the explicit rating of an item as an additional implicit input signal. Besides the discussed methods, many ways to hybridize explicit and implicit feedback have been proposed in the literature. Some focus on the specifics of certain domains, e.g., the music domain [48, 59], TV programs [4, 127] or web pages [129]. Others propose new techniques to combine the different types of feedback, for example, when using matrix factorization [75, 97].

14.4.4.1 Hybrid Neighborhood and MF Models. In application domains where explicit ratings are available, matrix factorization (MF) techniques can nowadays be seen as the state-of-the art for efficient and accurate rating prediction. In [60] Koren proposes to combine classic neighborhood models and MF for explicit ratings with implicit feedback. To that end, four hybridization strategies are introduced that build on each other: (1) a neighborhood model, (2) Asymmetric-SVD, (3) SVD++ and (4) an integrated model.

The first model is based on the classic way of predicting a rating for a user, e.g., by aggregating the ratings of similar items weighted by their similarity $\hat{r}_{ui} = \sum r_{uj} \cdot sim_{ij}$. The complete neighborhood model is defined as follows:

$$\begin{aligned} \hat{r}_{ui} &= \mu + b_u + b_i \\ &\quad + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_u - b_j) w_{ij} \\ &\quad + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij} \end{aligned} \tag{14.12}$$

Besides the overall average rating μ and the user and item biases b_u and b_i, the *neighborhood model* includes all explicit $R^k(i;u)$ and implicit ratings $N^k(i;u)$ of user u for the k nearest neighbors of item i, see Formula 14.12. For each item-item combination of i with its neighbors j, the sum is weighted with the factors w_{ij} and c_{ij} which model the strength of the relationship between i and j and are not given by a similarity function but learned in an alternating least squares learning phase discussed in [60].

The second model is *Asymmetric-SVD* in which the computationally expensive neighborhood calculation of Formula 14.12 is substituted by an MF approach and the rating prediction is therefore changed to:

$$\begin{aligned} \hat{r}_{ui} &= \mu + b_u + b_i \\ &\quad + q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_u - b_j) x_j \right. \\ &\quad \left. + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \end{aligned} \tag{14.13}$$

Instead of directly looking at all neighbors of item i to calculate a prediction, an "SVD-like" lower rank decomposition of the rating matrix is introduced. Compared to traditional SVD, e.g., $\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$, there are no user-wise latent factors p_u in this model. Instead, p_u is approximated and replaced with a term (between parenthesis in Formula 14.13) over all explicit $R(u)$ and implicit $N(u)$ ratings of user u. The parameters x_j and y_j are now latent item weights that are learned in the optimization process. As a side note, compared to p_u in classic SVD approach, the three model parameters, q_i, x_j, y_j, are not

user-dependent. Therefore, the model can directly predict ratings for new users without being re-trained.

The third model, *SVD++*, simplifies the *Asymmetric-SVD* model by reintroducing the latent factors p_u for each user u, but only for the explicit feedback. *SVD++* is defined as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \tag{14.14}$$

The final model combines both the *SVD++* and the *neighborhood model* into an *integrated model*. The underlying reason is that neighborhood models perform well when detecting localized relationships between few specific items but fail to capture the overall structure in a large set of ratings [60]. MF techniques, on the other hand, behave complementary. The hybrid approach is defined as:

$$\begin{aligned} \hat{r}_{ui} = \mu + b_u + b_i + q_i^T & \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \\ & + |R^k(i;u)|^{-\frac{1}{2}} \sum_{j \in R^k(i;u)} (r_{uj} - b_{uj}) w_{ij} \\ & + |N^k(i;u)|^{-\frac{1}{2}} \sum_{j \in N^k(i;u)} c_{ij} \end{aligned} \tag{14.15}$$

In their evaluation the authors used the Netflix dataset and generate implicit feedback by transforming the explicit ratings. They compare their methods against the classic neighborhood model $\hat{r}_{ui} = \sum r_{uj} \cdot sim_{ij}$ and SVD, and conclude that by adding implicit feedback, the recommendation accuracy can be significantly improved compared to the baselines. Also, *SVD++* performed better than *Asymmetric-SVD* when the implicit feedback was generated from explicit feedback. However, the authors state that for domains where implicit feedback is available, *Asymmetric-SVD* should in theory be more accurate.

14.4.4.2 Collaborative Feature-Combination. In [128] an approach to combine multiple (explicit and implicit) aspects of the user model was proposed. Classic CF approaches only take one type of rating data (explicit ratings) into account and are consequently challenged in cold-start situations. The proposed *collaborative feature-combination* recommender can help to deal with these challenges by considering existing implicit feedback – e.g. the navigation history of a user – if explicit feedback is not available. The general idea is to extend the single-category neighborhood calculation to multiple relevance-ordered feature dimensions. Then, the ranking score for the recommendation can be calculated as follows.

$$
\begin{aligned}
rec_{fch^*}(i,u,d_t,d_{rec}) &= \frac{\sum_{v \in N_u} score_{i,v}}{|N_u|} \\
&\text{with} \\
score_{i,v} &= sim_{fch^*}(u,v,d_t) \text{ if } i \in R_{d_{rec},v} \preceq R_{d_t} \text{ else } 0 \\
sim_{fch^*}(u,v,d_t) &= \sum_{d \preceq d_t} w_d \times cos(\overrightarrow{R_{d,u}}, \overrightarrow{R_{d,v}}) \\
cos(\overrightarrow{a}, \overrightarrow{b}) &: \text{cosine similarity} \\
R_{d,u} &: \text{rating vector for feature dimension } d \text{ and user } u \\
R_{d_t} &: \text{threshold feature dimension} \\
w_d &: \text{feature dimension weight}
\end{aligned}
\tag{14.16}
$$

In this equation, the recommendation score is the average of the weighted cosine similarity $score_{i,v}$ over all users N_u. The similarity $sim_{fch^*}(u,v,d_t)$ is calculated as a weighed combination over the feature dimensions, e.g., the implicit feedback of observed *buy*, *context*, *view* or *navigation* actions. In addition, the feature dimensions are ordered, for example by their predictive performance, i.e., $buy \prec context \prec view \prec navigation$. For the recommendation, a *threshold feature dimension* R_{d_t} has to be specified and the algorithm only uses feature dimensions that have a higher predictive accuracy than the threshold dimension. For example, by using the dimension *context* as the threshold, only implicit feedback of *buy* and *context* actions is included and the other (less meaningful) feature dimensions *view* and *navigation* are excluded in the calculation. The approach is therefore capable to gradually include different types of implicit feedback signals in the prediction model.

14.4.4.3 Bayesian Adaptive User Profiling. Similar to the collaborative feature combination approach, the authors of [134] propose a method to avoid the cold-start problem by simultaneously taking explicit and implicit feedback into account to model the user profile in a hierarchical Bayesian approach. Initially, there is only little explicit feedback available. Therefore, for new users, the model automatically focuses on the "cheap" implicit feedback and the collaborative information gathered from other users. The authors use a general Bayesian model to formalize this as follows:

$$
\begin{aligned}
f^u &\sim P(f|\theta) \\
y &= f^u(x) \\
&\text{with} \\
f^u &: \text{model of user } u \\
x &: \text{item} \\
y &: \text{rating}
\end{aligned}
\tag{14.17}
$$

From a general perspective, the user model is a function f^u for each user u that estimates a rating y for each item x and is modeled as a prior distribution on some parameters θ. In addition, the user model is personalized by learning

from a sample dataset D_u for each user that consists of item/rating-pairs. With Bayes' Rule, the general model can be extended to:

$$\begin{aligned} P(f^u|\theta, D_u) &= \frac{P(D_u|f^u,\theta)P(f^u|\theta)}{P(D_u|\theta)} \\ &= P(f^u|\theta)\prod_{i=1}^{N_u}\frac{P(f^u(x_i^u)=y_i^u|f^u)}{P(f^u(x_i^u)=y_i^u|\theta)} \\ &\text{with} \\ D_u &= \{(x_i^u, y_i^u)|i=1\ldots N_u\} \\ N_u &: \text{number of training samples for } u \end{aligned} \tag{14.18}$$

For each user u, the belief about the user model is also based on the training data D_u. Equation 14.18 shows that the user model depends both on the model's prior probability $P(f^u|\theta)$ and the data likelihood given the user model f^u. If the number of training samples N_u for a user is small, i.e., there is little explicit feedback available, the prior probability based on the observed behavior and other users is the major contributor of the final model. For the prior, the authors use a hierarchical Gaussian network which is further discussed in [134].

14.4.4.4 Reciprocal Compatibility. In [99], explicit and implicit feedback is used in the domain of online dating as a two-step approach. The explicit feedback, which consists of features like age and body type that the user prefers, is used to filter the possible recommendation results. The ranking of user profiles, on the other hand, is based on the implicit feedback – viewing user profiles, sending and replying to messages – by calculating a "reciprocal compatibility score". This similarity measure is calculated as follows:

$$\begin{aligned} recip\_compat(u,v) &= \frac{2}{compat(u,v)^{-1}+compat(v,u)^{-1}} \\ &\text{with} \\ compat(u,v) &= \sum_{i=1}^{n}\sum_{j=1}^{k_i}\frac{f_{u,i,j}}{k_i}\times P(v,i,j) \end{aligned} \tag{14.19}$$

Here, $P(v,i,j)$ indicates that some feature A_i (e.g., *body type*) has a certain value a_{ij} (e.g., *slim*) in the profile of user v. The factor $f_{u,i,j}$ is the implicit preference of user u for that features value a_{ij}, e.g., the number of times the user viewed the profile of a *slim* user. Therefore, this approach uses the observed user behavior to weight the preference of explicitly given features.

14.5 Evaluation Aspects

In this section, we will discuss aspects related to the accuracy evaluation for implicit feedback situations. We will limit our discussion to settings in which we only have "positive" user-item interactions (unary feedback) available for learning and evaluation, i.e., there are no negative signals and no graded positive feedback signals. See also Chapter 10 of this book [55] for a more detailed discussion of evaluation metrics for ranking tasks. Furthermore, we assume that we apply the usual procedure of splitting the available data into training and test data and perform cross-validation, a sort of repeated subsampling.

As in this chapter previously discussed, the problem is that we usually cannot know if entries in the user-item interaction matrix are missing because the user did not like the item (and therefore, e.g., did not purchase it) or the user was simply not aware of the item.

14.5.1 Recommendation as a Classification and Ranking Task

One basic functionality of a recommendation algorithm is to classify each item into one out of two categories, i.e., predict if a user will like it or not. If we treat the empty cells as containing zeros and the others as ones, we can use any classification technique and compare their performance using standard measures from the literature. Given the outcome of the classification process, we can compare the results with the "ground truth" in the test set and categorize each test outcome in one of four groups: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) as shown in Table 14.4.

Table 14.4 Contingency table of classification outcomes.

	Relevant	**Non-relevant**
Recommended	True Positives	False Positives
Not recommended	False Negatives	True Negatives

Assessing Unranked Recommendations. Based on this categorization, we can compute the True Positive Rate ("How many of the *ones* in the test set have been correctly predicted as such", a.k.a. Recall), the True Negative Rate and so on. In recommendation settings, the measurement is typically done with a predefined list length and the Information Retrieval measures Precision and Recall. Precision measures how many of the elements placed into a top-k recommendation by an algorithm were truly "relevant". The Recall indicates how many of *all existing relevant items* made it into the top-k list. The values for Precision and Recall depend on the length of the top-k list. The Recall value will typically

increase with longer list lengths while Precision might decrease at the same time. The F-measure can therefore be used as weighted combination of Precision and Recall using, e.g., the harmonic mean of the values.

Assessing Ranked Recommendation Lists. In most recommendation and information retrieval scenarios, the first few elements of the top-k lists receive the most attention from users. Precision and Recall do not take the position of "hits" (i.e. True Positives) into account, but we know that changing the list length will influence their values. A good classifier should therefore be able to place the hits at the top of the ranked list and will lead to high Recall values already for short lists. A poorer classifier might have more False Positives at the top and achieves similar Recall values only at longer list lengths (at the cost of lower Precision).

To assess how different classifiers perform using different thresholds (list lengths), Precision-Recall curves can be used, where Precision is plotted as a function of Recall. Receiver Operating Characteristic (ROC) curves are a related visual approach, which can be derived based on the position of the True Positive Rate and the False Positive Rate in the ROC space under varying threshold levels.

In order to make the comparison of such curves easier, researchers use the measures Mean Average Precision as the average Precision at each threshold level or the Area Under the Curve (AUC) for ROC curves[5]. A discussion of the similarities and differences between ROC curves and Precision-Recall curves can be found in [22]. A particular aspect according to this work is that ROC curves tend to overestimate the performance of a classifier when compared to the usage of Precision-Recall curves.

Single-Item Rank Position Measurements. The *Mean Reciprocal Rank* (MRR) measure considers the position of the first hit in a ranked list of items. Its value is computed as the multiplicative inverse of the position (rank) where a hit occurred, e.g., if the hit occurred at the third position, the MRR will be 1/3. The MRR was used as an optimization goal for example in the CLiMF method [114].

Another evaluation variant for the top-k recommendation was used in [20] and [60]. The idea of the protocol is to evaluate each relevant item of each user in the test set individually by combining it with n (e.g., $n = 100$) other items for which the ground truth is not known, i.e., for which no rating exists from the given user. The task of the recommender is to rank these *n+1* items. Recall is then determined at a certain list length k and can be zero or one, depending on if the test set item was in the top-k list or not. Precision is defined as *Recall/k* in this setting. To make the measure independent of the list length, sometimes the *percentile* rank of the hit is reported (*Mean Percentile Rank*), see [17] or [41].

[5] The BPR-OPT criterion used in the previously described BPR method has a close correspondence to the AUC measure.

14.5.2 Domain-dependent Evaluation Approaches

The most popular measures in the Recommender Systems literature (Precision, Recall, RMSE/MAE) [47] have the advantage that they are domain-independent ways of comparing different classifiers or recommendation algorithms. Unfortunately, the choice of the specific evaluation measure often seems arbitrary in the literature even though it clearly depends on the domain which measure should be optimized, e.g., if the problem is to find "one good item" or "find all good items" etc. [38]. The RMSE, for example, has been heavily used since the Netflix prize even though it is not fully clear if high prediction accuracy comes with high user satisfaction or increased sales.

Problem-specific measures in implicit feedback settings have, e.g., been proposed in the domains of e-commerce recommendations and music playlist recommendation. The ACM RecSys 2015 Challenge, for instance, was based on time-stamped and sessionized user activity logs from a online store and the participants had to accomplish two tasks. For each session in the test set, the task was to predict if a visitor will purchase an item and if so, which item will be bought. The used evaluation measure was then a combined score which (a) takes into account how often a session with a purchase action was properly predicted and if the purchased item was correctly predicted and (b) penalizes all wrong predictions of sessions with a purchase action. The winning strategy in the contest used a two-phase classification approach that in the first step identifies sessions that will probably include at least one purchase and in the second step detects individual items that will likely be bought in these sessions. A high accuracy of the task was achieved by considering time-based aspects, i.e., the time and day when users tend to do a purchase.

Another evaluation protocol when using time-stamped activity logs was proposed in [44]. The specific research question was to estimate how fast different recommendation approaches adapted their buying proposals to the short-term interests of customers. The protocol therefore includes a mechanism to incrementally "reveal" the most recent clicks to a recommender. Different strategies were evaluated on an e-commerce dataset and the results indicated that considering short-term interests and the most recent user actions are crucial to achieve high accuracy, but also that having long-term preference models is important. In both discussed e-commerce settings, user sessions and activity time-stamps are crucial for being able to compare different recommendation strategies in a more realistic way.

In the music domain, the Recall was, for instance, used to compare different strategies of generating playlist continuations (next-track recommendations) , see [11] or [36]. The input to the recommendation algorithms were hand-crafted playlists shared by users. The specific setup is to predict the very last track of each playlist in the test set. Differently to standard approaches, very long list lengths were used in the evaluation process (e.g., 300 items). In particular for the problem of playlist generation, several other measures were proposed in the literature, including the *Average Log Likelihood* that an algorithm will produce

a "natural" playlist as well as other list-based measures that aim to determine quality factors like homogeneity or the smoothness of transitions, see [11].

14.5.3 Discussion

Offline evaluation designs as discussed above in general have a number of limitations independent of the type of input data, i.e., implicit or explicit feedback, such that we cannot be sure if optimizing for high accuracy leads to the desired effects from an application perspective or that accuracy measures do not capture other possible quality factors like diversity or novelty. When only implicit feedback signals are available, some further aspects should however be considered.

The Lack of the Ground Truth for Negative Feedback. The classic IR measures Precision and Recall were designed with the idea that we can organize the prediction outcome in the contingency table (Table 14.4), i.e., we know the expected outcome for the ranked items, and research was often based on manually labeled documents. In RS applications the ground truth for most items is however unknown, which means that in particular the Recall measure has to be interpreted with care [38] and – depending on the specific way of measurement – can only be considered as a lower bound of the accuracy of a recommender in reality [11, 114].

To evaluate their implicit-feedback algorithms, some researchers use common rating-based datasets (e.g., from MovieLens) and convert them to binary ratings using an arbitrary rating threshold[6]. Every rating below the threshold is considered a negative feedback. Although such an evaluation can give us some insight about algorithm performance for a two-class explicit-feedback setting, it might not be truly representative for the performance of algorithms when only positive feedback is available. Furthermore, depending on how the items with missing ground truth are counted when determining the values for Precision and Recall, completely different algorithm rankings can result from an comparative evaluation [46].

Finally, when converting explicit feedback to binary feedback, also the standard prediction accuracy measures (RMSE, MAE) can be applied in particular for cases where the recommendation algorithms do not only output binary predictions. Again, since the underlying data is not truly a one-class collaborative filtering problem, the results of such an offline comparison might be representative for the true performance of the different techniques.

In the other direction of implicit-explicit mappings, one could try to transform different forms of implicit feedback to different levels of explicit feedback. A "purchase" could be a five-star rating, an item view correspond to three stars etc. Then all forms of rating-based algorithms can be applied and evaluated in

[6] This was for example done for the evaluation of the implicit-only algorithm BPR, see `http://www.mymedialite.net/examples/item_recommendation_datasets.html`

terms of error measures such as the RMSE. Again, the outcome of the evaluation process might depend on the particular encoding of the implicit signals. Furthermore, the number of available ratings at each level might be very different, i.e., there might be orders of magnitude more view events than purchases, such that constantly predicting a "three" might be a competitive strategy when error measures are used.

Popularity Biases. Even when explicit rating information is available, we often see that the distribution of the ratings is quite skewed and the more positive ratings dominate. Even more, research suggests that ratings are not missing at random [77], which can lead to undesired biases in the recommendation models. At the same time, we often see a very skewed distribution regarding how many ratings an individual item received, i.e., there is typically a long-tail distribution in which a small set of all items obtains a large fraction of all existing ratings.

One particular problem in that context is that in implicit feedback settings there is a huge number of "negatives" when we try to compute Precision and Recall. If we rank ten thousands of items, the chances of placing one of the few dozen "true positives" in a top-10 list are very low[7]. In some works, researchers therefore rely on very long top-k lists to avoid that they have to compare very tiny Recall values (like 0.003 vs. 0.004) [11, 36], or report (cumulative) probability percentages and use subsets of items to be ranked [61].

Generally, as a result, recommending the most popular items to everyone can be a baseline that is hard to beat [20]. Being better than such a simple approach in terms of Precision and Recall can often only be achieved by using algorithms which themselves can exhibit a strong popularity bias like BPR [46]. In reality, recommending only popular items can certainly be of limited value for the user. In some works like [114], researchers consider the n most popular items as being irrelevant for the user and do not "count" them in their evaluation. Although this might be reasonable from a practical perspective, the choice of the top n elements to be ignored appears somewhat arbitrary in particular if it does not consider the individual user's knowledge about certain items.

14.6 Case Studies And Selected Technical Approaches

In this section, we will discuss case studies and selected technical approaches from different domains in more detail to obtain a deeper understanding of approaches based on implicit feedback and how researchers deal with the specific challenges. The case studies are selected from the e-commerce and music application domains as in these domains implicit feedback is often prevalent. In addition, we present technical approaches for implicit feedback algorithms that address the popularity bias of implicit feedback and the support of graded feedback.

[7] Many more of the top-ranked elements might be relevant for the user, but no explicit information is given.

14.6.1 Recommending based on Activity Logs in E-Commerce

We will focus on two recent approaches that use the activity logs of Zalando, a larger European online retailer for fashion products, as described in [119] and [44]. We chose these two works as they are based on a real-world dataset[8] that contains information that is (a) typically available for many online shops and (b) corresponds to what companies might share with researchers as no sensitive information is contained in the logs. Furthermore, the log contains *all* user interactions for a given time period[9] and is not limited to a particular user group, e.g., *heavy* users. The social aspect when generating recommendations in this setting is the collective behavior of the website users which is analyzed to identify patterns in the navigation and buying behavior.

14.6.1.1 Data Aspects. The user activity log contains more than 24 million recorded user actions of different types (item views, purchases, cart actions). Most of the actions – about 20 million – are item views and about 1 million actions correspond to purchases. The user actions are related to more than 1.5 million *sessions*, which comprise sequential actions within a certain time frame. Each log entry comprises a limited amount of information about the item itself like the category, price range or color. The actions were performed by about 800.000 anonymous users. The catalog of products (including product variants) appearing in the log is huge and consists of around 150.000 items.

The dataset exemplifies several of the challenges mentioned in Section 14.2.3, including, e.g., the abundance of data[10], the general sparseness with respect to the available purchase data as the majority of users has never made any purchase, or the problem of the interpretation of the strength of the different signals.

On the other hand, such datasets allow us to perform analyses and design algorithms which are much closer to the demands of real-world recommendation systems than the non-contextualized ex-post prediction of missing entries in a user-item rating matrix, which is the most common evaluation setup in research [47].

14.6.1.2 Topic Detection for User Sessions. In [119], the authors present an approach to automatically infer the "topic" or short-term shopping goal for the individual user sessions. The proposed approach for topic detection is based on Markov Decision Processes (MDP) and can be easily transformed to serve as a topic-driven recommendation technique or MDP-based recommender system [111].

The basic idea of their approach is to view each session as a sequence of item attributes as shown in Figure 14.6. The example for instance shows that the

[8] The data is not publicly available.

[9] The data was sampled in a way that no conclusions about visitor or sales numbers can be drawn.

[10] The data sample was taken within a very limited period of time.

user has only viewed items from the category "shirt". The shirts had however different colors and different price ranges. The general topic (shopping goal) to be inferred is shown on the right hand side of the figure, i.e., the user looked for dark-colored shirts for women in any price range.

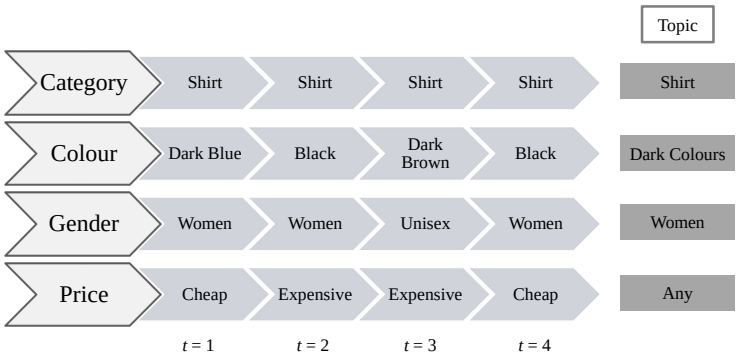

Fig. 14.6 Viewing a session as a sequence of attributes [119].

Approach. Technically, the idea is to model the topic detection problem as a reinforcement-learning problem based on Markov Decision Processes. The observed sequences of actions – in this case sequences of item features – are therefore used to train a model to predict the most likely next observation (state). The distribution of the item attribute values in the user session are considered the *topic* of the session which can then be leveraged in the recommendation process. The learned models are strictly session-dependent, i.e., no long-term profile of the individual user is built in this approach.

Relying on MDPs for the recommendation task was done, although in a different form, for example in [111]. The particular challenge however lies in the computational complexity of such an approach given the huge amounts of items and possible states. In [119], this scalability problem is addressed by using *factorized* MDPs. They rather model sequences of item attribute values than sequences of observed interactions with items and learn such fMDPs independently for each attribute. Furthermore, an approximation technique is used in the optimization phase to avoid scalability problems in terms of memory requirements.

As a result of the (approximate) optimization process we obtain probabilities which express the most likely next observed attribute values. This information can be used to extract the topic of the session as well as to rank items based on their particular item features.

Results. In their empirical evaluation, the authors first compare different strategies for topic detection. The results show that their method is highly accurate in predicting the topic (around 90%, depending on the length of the observed history) and much better than the compared baselines, among them a simple Markov process based on the frequencies of item clicks.

Second, a comparison was made for the recommendation tasks where the baselines include both models that rely on the long-term user profile and latent factor techniques as well as more simple baselines that recommend popular items or items that are feature-wise similar to the last viewed item. As an evaluation measure, the *average rank* (position) of the correct item in the recommendations was used. The results show that the proposed MDP-based method is better than the collaborative filtering (CF) methods which rely only on longer-term models. In addition, also the simple contextualized baseline methods are better than the CF methods.

Discussion. From a general perspective, the experiments in [119] show that the consideration of short-term shopping goals and the sequence of the observed user actions can be crucial for the success of recommendation systems in real-world environments. Assessing the true value of the final recommendations unfortunately remains challenging as even the best performing method only lead to an average rank of 15,000 (due to the large item assortment in the shop).

The results also indicate that optimizing for long-term goals alone as done in the state-of-the-art baseline methods can be insufficient. Overall, at least in the e-commerce domain, using implicit feedback data with time information might help us to develop models which are much closer to real-world requirements than models that generate time- and situation-agnostic predictions for missing items in the rating matrix. Furthermore, the work highlights scalability limitations of existing approaches when it comes to deal with real-world datasets. For the first set of experiments, the authors merely used a few percent of the available data to be able to perform the optimization. For the larger dataset, unfortunately no information is provided on computation times for model building and generating recommendations.

A related case study can be found in Chapter 16 of this book [12] for context-aware recommendations of places. There, a post-filtering approach according to the user's short-term goals is employed to create an intention-based ranking of nearby relevant locations.

14.6.1.3 Evaluating the Combination of Short-Term and Long-Term Models. The discussion in the previous section indicated the importance of generating recommendations that consider the recent short-term user intent while solely exploiting long-term preference models might be insufficient. In fact, many of the recommendations of popular e-commerce sites like Amazon.com are either simply reminders of recently viewed items or recommendations that are connected to the currently viewed item ("Users who viewed ... also viewed ...").

In [44], the authors aim to quantify the effectiveness of such comparably simple recommendation strategies and furthermore analyze the possible benefits of combining them with optimized long-term models. One further goal is to assess how quickly the different strategies adapt their recommendations after the most recent user action in a session.

Evaluation Protocol. Since standard evaluation setups in the research literature do not cover situations in which time-ordered session logs are available, the

authors propose a parameterizable and domain-independent evaluation protocol as shown in Figure 14.7.

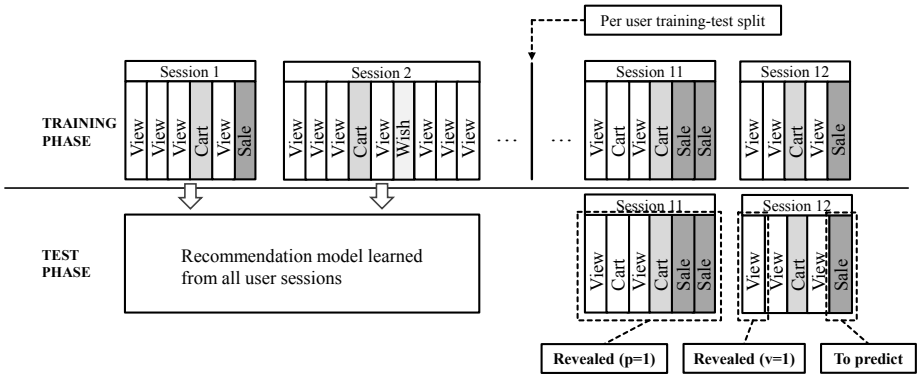

Fig. 14.7 Proposed Evaluation Protocol [44].

The general idea is to split the data as usual into training and test data while maintaining the order of the log entries. The task in the test phase is then to predict which item will be purchased in a session of the training set. In contrast to similar protocols, e.g., the one used in the ACM RecSys 2015 Challenge, the idea is now to vary the amount of information that a recommender is allowed to see from the *current* and *previous* session. In one setup, we could for example reveal the first 2 item views of the current session and all user actions of the preceding session. Using this extra information, it is for example possible to assess the effectiveness of a strategy in which the most recently viewed items are displayed. As a success measure, the Recall can be used which indicates if the purchased item was in the top-k list.

Algorithms and Results. A number of different algorithms were used, including the one-class CF method BPR described in Section 14.4.2.3 as well as the more recent Factorization Machines approach of [100]. These long-term preference modeling approaches were then combined with a number of short-term adaptation strategies, including approaches which recommend (a) the most recently viewed items, (b) items that are content-wise similar to those viewed in the current context, (c) generally popular items, or (d) items that co-occurred with the currently viewed ones in past transactions. Combinations of the different short-term strategies were tested as well.

Similar to the findings reported in [119], the results show that standard CF methods like Factorization Machines do not perform well at all in this evaluation setup and only the BPR method, which has a comparably strong popularity bias, outperforms the popularity-based baseline when no context information is available.

All short-term adaptation strategies on the other hand however immediately led to a strong increase in terms of the Recall even when a weak baseline strategy

was used and only the first two item views in a session were revealed. Although the comparison of context-agnostic long-term models and the short-term strategies is in some sense "unfair" as a few more user actions are known to the short-term strategy, the strong increase in accuracy helps us to quantify the importance of the adaptation process.

The best-performing method in the end was a hybrid technique which used BPR as a baseline and adapted the recommendation lists by favoring both recently viewed items as well as items whose features are similar to those that were viewed in the current session[11]. In absolute numbers, the Recall of the baseline method of 0.40 was increased to 0.66 through the hybrid method for a configuration in which only the first two item views of the current session and the last two preceding sessions were revealed.

Discussion. Although the short-term adaptations in the experimental analysis were effective, the results also show that the choice of a strong baseline and the capability of understanding the user's long-term preferences are important. On the other hand, while the results of the log-based analysis emphasize the importance of considering short-term interests, it is not fully clear whether the "winning" models fulfill the business goals of the shop owner in the best possible way. The BPR method, for example, can exhibit a comparably strong tendency of recommending popular items and is probably not very helpful when the goal of the recommendation component in a shop is to guide the customers to long-tail items or to help them discover additional or new items in the catalog.

Reminding users of recently viewed items shows to be very effective, e.g., because users might have a tendency to postpone their buying decisions for at least another day in order to sleep on them. However, while the strategy leads to good results in terms of the Recall, it is unclear if the recommendations generate any additional revenue for the shop owner.

In the work in [44], user actions like "add-to-wishlist" or "put-in-cart" were not considered and more work is required to understand (a) how to weight these user actions in comparison to, e.g., view actions and (b) whether or not it is reasonable from an application perspective to remind users on the items in their carts or wishlists.

14.6.2 Next-track Music Recommendations from Shared Playlists

Many implicit feedback techniques were developed in the context of media services where the media consumption behavior of users was monitored and analyzed to understand their preferences. A special form of a social action for the users on several media platforms like Last.fm or YouTube is that they can share so-called *playlists* (ordered sequences of music tracks or videos) with others.

[11] The importance of feature-based similarities was also the basis in [119].

In this section, we will highlight some aspects of music playlist generation and in particular approaches that rely on playlists shared by other users. The particularity of this recommendation problem are that

1. shared playlists represent a form of feedback which is not related to one single item but to the whole recommendation list,
2. and that the recommendation outcome is usually not a list of items where the user should find at least one relevant element but rather a list of items which should be sequentially consumed by a user.

A shared playlist has characteristics of explicit and implicit feedback. Regarding the individual tracks, one can usually safely assume that the user who shares the list likes at least most of the individual tracks in the playlist. In addition, the organization of the tracks in a playlist is also an indicator that the collection of tracks leads to a certain listening experience for the user because playlist (mix) creators typically have a certain underlying theme or goal in mind when selecting and ordering the tracks [11].

In this section, we will address the problem of "next-track music recommendation" with a particular focus on approaches that use shared playlists as a social-based input. The general problem setting is that we are given a sequence of recently listened tracks or a "playlist beginning" and the task is to create a continuation. This problem corresponds to generating a virtually endless jukebox or radio station for a given seed track or artist.

The problem can be considered as consisting of two subtasks. The first one is to identify tracks that generally match the playlist beginning, e.g., in terms of the musical genre. The second task is to bring the tracks into a certain order, e.g., to achieve smooth transitions between the tracks or to avoid that multiple tracks of the same artist appear in the playlist.

14.6.2.1 Identifying the Right Tracks. Various techniques to select good tracks for a playlist continuation have been proposed in the past. They can be distinguished, for example, by their underlying objective when selecting the tracks.

Popularity-based Approaches. In [78], McFee et al. propose a very simple baseline method in the context of the Million Song Dataset (MSD) challenge[12] called "same artists – greatest hits (SAGH)". The simple idea is to take the set of artists appearing in the listening history of the user and predict that users will listen to the greatest hits (most popular tracks) of the same artists in the future.

A comparison with a baseline based on recommending the generally most popular tracks and the BPR algorithm shows that the simple strategy outperforms the other techniques on different accuracy measures like Mean Average Precision, Precision or the Mean Reciprocal Rank. This suggests that popularity and concentration biases have to be considered. In their dataset, half of the user-track interactions were related to only 3% of all the tracks.

[12] `http://labrosa.ee.columbia.edu/millionsong/challenge`

A straightforward extension to the SAGH scheme was proposed for playlist continuations in [10] called "collocated artists – greatest hits" (CAGH). Instead of only playing the greatest hits of the artists in the playlists, the top hits of similar artists are recommended, where the similarity between two artists can be computed based on the co-occurrence of the tracks of the artists on the existing playlists.

Co-Listening Events and Nearest-Neighbors. In the Million Song Dataset challenge the task was to continue the listening histories of thousands of users in the test set for which the first half of their history was revealed. The challenge indicates that relying on co-listening events is a particularly well-suited strategy, i.e., quite simple collaborative filtering techniques that look for users who (frequently) listened the same tracks in the past, see e.g., [30], or search for track co-listening patterns that occurred often in the training data worked quite well while matrix factorization approaches did not.

While the MSD challenge focused on full listening histories, playlist generation techniques are often designed to continue an individual given playlist beginning without knowledge about the creator's listening history. The focus in contrast to, e.g., the MSD challenge is to predict the immediate next tracks. For such cases, different forms of considering track co-occurrences were proposed and evaluated in the literature, including for example Sequential Patterns [10], latent Markov Embeddings [81] and *k-nearest-neighbor* (kNN) methods [36].

Using the kNN method with large neighborhood sizes is particularly effective. The basic idea when applied to the playlist generation problem is to encode all playlists as binary vectors, in which the vector elements are the available tracks and a vector contains a 1 in case a track appeared in the playlist beginning. Based on this encoding, similar playlists can be found by computing the cosine similarity between the playlist beginning and all training playlists. The *score* of the recommendable track can then be computed based on how often it appears in similar playlists, weighted by the similarity score.

Using Additional Information. The construction of a playlist can be governed by different guiding "themes", which can be related to the artists or genres of the tracks, musical features like tempo or loudness, as well social aspects like the popularity of the tracks.

In [36], the idea is to try to identify the "topic" of a playlist based on the social tags attached to the tracks. Such tagging information can for example be obtained from the Last.fm music platform. The particular idea in [36] is to determine a set of latent topics for each track and then mine the training dataset of playlists for topic transitions. These patterns are then combined with a kNN method and slight increases on the hit rate were observed when small neighborhood sizes for the baseline were used.

A multifaceted track scoring approach was proposed in [45], where again the kNN method was used as a baseline which can be combined with a number of additional scores. Each additional score expresses the match of each track according to a certain criterion, e.g., if the track matches the tempo of the

playlist beginning, if it has the same popularity, or if the track has similar tags attached as tracks that the user frequently puts into the playlists. This last aspect can therefore be used to introduce a personalization aspect into the playlist continuation process.

Discussion. Most of the presented approaches were (at least) evaluated in terms of Recall. A common setup is that the last track of each playlist in the test set is hidden. Recall is then measured by determining if the hidden track appears in the top-n list returned by the "playlister". The results for three different sets of playlists shared by users presented in [45] indicate that using a combination of features (co-occurrence, "content" based on social tags, etc.) can be a promising strategy to obtain high accuracy.

14.6.2.2 Generating Coherent Continuations. Finding suitable tracks to continue the recent listening history or playlist is often only one part of the problem. Achieving a high hit rate (Recall) might therefore not be sufficient as an optimization goal because other criteria like track transitions or homogeneity can be important, too [130]. Furthermore, already a few "wrong" tracks placed in the playlist continuation can have measurable impact on the quality experience of the listener [16].

In the literature, for example, a number of optimization-based approaches have been proposed – see [11] for an overview – which try to create playlists that obey certain user-defined constraints regarding, e.g., the start and end track, the number of different artists etc. A problem of some of these approaches lies in their limited scalability as the optimization problem becomes very complex given that there are usually millions of possible tracks that can be included.

In [130], a method was proposed that combines multiple algorithms with the goal of obtaining a high accuracy while at the same time observing other desired quality factors like diversity, novelty, and in particular serendipity.

Similar goals were in the focus of the work presented in [45], where the authors present a generic optimization-based re-ranking procedure that can be used to balance different optimization goals. The technique takes an accuracy-optimized playlist continuation as input and systematically exchanges top-ranked elements with lower-ranked elements in case the lower-ranked element represents a better "match" for the most recent playlist, e.g., in terms of the tempo or the general topic. An empirical evaluation on different datasets showed that the re-ranking technique not only leads to more homogeneous playlist continuations that better match the playlist beginnings, but also can help to further increase the accuracy of the track selection process.

14.6.3 Considering Application-Specific Requirements in BPR

BPR is a state-of-the-art ranking algorithm for one-class collaborative filtering situations (Section 14.4.2). Since its original presentation in [101], several variations and extensions were proposed in the literature, e.g., to make the algorithm better suited for certain application requirements. In this section, we will look at some of these proposals in more detail. Among others, we will discuss an approach to counteract the popularity bias of the algorithm and present algorithm extensions to deal with graded relevance feedback.

Besides the discussed methods, multiple other enhancements to BPR were proposed in the literature. The improvements are for example related to including the temporal information, the social connection or the item taxonomy [25, 50, 102]. Some approaches also extend the two-dimensional user-item perspective of BPR towards additional dimensions [62, 76, 103]. In terms of the pair-wise item-item relations, there are some approaches that introduce the concept of group-wise relations [89, 90].

14.6.3.1 Dealing with the Popularity Bias. Some one-class collaborative filtering algorithms discussed in Section 14.4.2 use different strategies to create artificial negative feedback signals. In most cases, some kind of weighting or sampling scheme is used to derive negative feedback from the structure of the observed interactions. In the wALS, sALS-ENS, Random Graph and BPR approaches, the created negative examples were chosen in a way that was inversely proportional to the popularity of the items, i.e., the algorithms assume that popular items are more acceptable.

While in general this assumption seems plausible, it can however lead to a popularity bias in the recommendations, i.e., the algorithms tend to recommend popular items to everyone. In [46], the authors compared a number of recommendation techniques regarding popularity and concentration criteria and showed that BPR strongly tends to focus on the most popular items. Although popularity-biased recommendations can lead to high values in terms of Precision and Recall [46], the bias might be undesired in specific application settings.

In BPR, the popularity bias emerges from the specific way the algorithm takes samples to learn the preference relations. As discussed in 14.4.2, the BPR algorithm optimizes the set of model parameters Θ with a stochastic gradient descent procedure by randomly sampling triples (u, i, j) from D_S. The distribution of the observed (positive feedback) signals is typically non-uniform, i.e., the popularity of the items has a long-tail shape. As a result, sampling the triples randomly from all observed interactions leads to many sampled triples (u, i, j) where the item i belongs to the more popular items. The item j, however, is randomly sampled over all items and thus more likely to be from the long-tail of unpopular items. As a result, the gradient descent algorithm updates the model parameters with many triples (u, i, j) that contain a (popular, unpopular) item pair and therefore BPR favors popular items in the recommendation step. In

[46], an adapted sampling strategy was introduced that counters the popularity bias of BPR.

Approach. Instead of applying random uniform sampling, a modified distribution function ϕ is used to sample the triples (u, i, j). The non-uniform sampling with a distribution ϕ biases the sampling probability of the items i in a way that more unpopular items are sampled. As a result, the model is updated more often with tuples (u, i, j) where i is less popular. Figure 14.8 shows the shape of a sampling function ϕ that can be used to sample the positive feedback signals of items i (dashed line) in comparison to the popularity distribution of the item space (solid line). Function ϕ is a monotonously decreasing distribution function that samples more popular items with a lower probability.

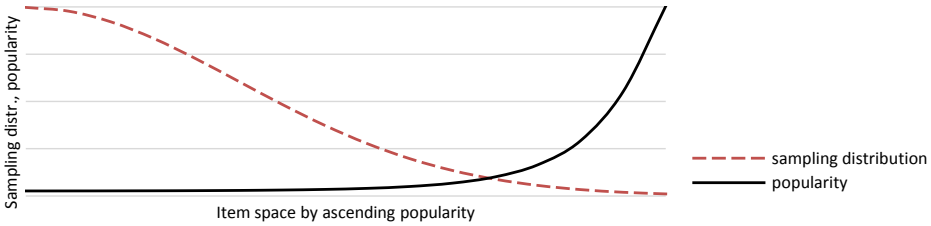

Fig. 14.8 Sampling distribution for items i with positive feedback compared to their popularity. The x-axis represents the items space by ascending popularity. The y-axis contains the distribution of the item popularity (solid line) and sampling probability of function ϕ (dashed line). Popular items are sampled less frequently.

Different distribution functions for ϕ are possible and in [46] a normal distribution $\phi(\omega)$ with a mean of $\mu = 0$ and a standard deviation $\sigma = \frac{|L_u|}{\omega}$ is used. Here, $|L_u|$ is the number of rated items of user u.

$$\phi(\omega) = \mathcal{N}\left(0, \left(\frac{|L_u|}{\omega}\right)^2\right) \tag{14.20}$$

The strength of the counter-bias when sampling can be chosen by varying the breadth of the function with the parameter ω, e.g., increasing ω leads to a narrower distribution $\phi(\omega)$ and less popular items are sampled for i. On the other hand, setting $\omega < 1$ leads to a more uniform selection of items.

Results and Discussion. The proposed sampling method [46] was compared with the original implementation by Rendle et al. [101] on the MovieLens and Yahoo movie datasets. The results show that there is an (expected) trade-off between recommendation accuracy and the popularity bias. When increasing the width ω of the sampling distribution, thus focusing on more unpopular items, the *average popularity* and the *Gini index* (to measure the concentration of recommendations) decrease, which means that the algorithm no longer recommends the same set of blockbusters to everyone. At the same time, however, also Precision and

Recall decrease, but at a much lower rate. A 5% decrease in accuracy can for example be traded in for a 10% reduction of the overall popularity of the recommended items. The actual size of the desired effect can be determined based on the specific application requirements.

14.6.3.2 Supporting Graded Relevance Feedback. In many domains, implicit feedback occurs not only as a binary indicator but in a graded form. In the e-commerce domain, for example, different observable user actions like item views or purchases are interest indicators of different strength. The repeated consumption of items in an online media service is another indicator which should be interpreted as a stronger signal than a single consumption event.

In its original form, BPR only supports binary feedback but there are some extensions that allow different graded signal strengths to be considered when learning the preference relations.

BPRC. To that end, Wang et al. [122] introduced a confidence weight in the objective function of the BPR-Opt criterion. In their *BPR with confidence* approach, the confidence score originates from the problem setting of recommending social network messages and is calculated based on the difference of reception times of two messages. Thus, the optimization criterion is extended as follows to BPRC-Opt:

$$\begin{aligned} \text{BPRC-Opt} &= \sum_{(u,i,j)\in D_S} \ln \sigma(c_{uij}\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2 \\ &\text{with} \\ c_{uij} &= \frac{1}{t_i - t_j} : \text{confidence weight} \end{aligned} \tag{14.21}$$

The confidence weight c_{uij} is the inverse of the difference between the reception times t_i and t_j of two messages. If the time between two messages is long, the confidence weight c_{uij} lowers their impact $\hat{x}_{uij}$ in the training phase of the model. Since the confidence values are given by the application setting, the optimization is analogous to BPR.

The approach was benchmarked against classic kNN, MF and BPR on a dataset from Sina Weibo[13], which is a similar microblog system as Twitter. The BPRC approach has the same recommendation characteristics as BPR but has a higher accuracy in terms of Precision and Recall. In addition, the authors report that the confidence-based method greatly outperforms many other algorithms in a cold-start scenario.

ABPR. In [91], a similar generalization of BPR for so-called "heterogeneous implicit feedback" has been proposed. This *Adaptive Bayesian Personalized Ranking* (ABPR) has the ability to model and reduce uncertainty for different types of observed feedback. In their work, the authors have a problem setting with

[13] `http://www.weibo.com`

two types of implicit feedback, transactions and examinations, i.e., item purchase events and item click events in an online store. A naive approach would be to assume that both types of user actions are equivalent positive implicit feedback. However, viewing a product page is not necessarily positive feedback, and using only the transactions would result in sparse training data. Therefore, like the confidence-extension for BPR [122], the ABPR approach assumes confidence weights for both types of feedback. The optimization criterion in thus extended to:

$$\begin{aligned} \text{ABPR-Opt} = & \sum_{(u,i,j) \in D_S} f^T_{uij}(c_{ui}, \Theta) + \lambda_E f^E_{uij}(c_{ui}, \Theta) - \lambda_\Theta ||\Theta||^2 \\ & \text{with} \\ f^T_{uij}, f^E_{uij} & : \text{estimation function for transaction, examination} \\ c_{ui} & : \text{individual confidence weight} \\ \lambda_E & : \text{global confidence weight} \end{aligned} \tag{14.22}$$

The optimization criterion now depends on both the estimation of the transactions f^T_{uij} and examinations f^E_{uij}. Furthermore, the impact of the examinations is controlled by a global confidence weight parameter λ_E and the individual confidence weights c_{ui} determine the impact for each triple transaction or examination triple (u, i, j). Compared to the BPRC approach where the messages had a time stamp that determined the confidence, in this setting the weights are not deduced directly from some meta-data and are instead determined in the optimization process. When there is a transaction for user u and item i in the training data, the confidence weight is assumed to be 1. Otherwise it is initially unknown and learned in the extended stochastic gradient descent algorithm discussed in [91]. This is in some sense similar to the graph-based approach for one-class implicit feedback discussed in Section 14.4.2.2.

ABPR was benchmarked on MovieLens and Netflix datasets. In terms of accuracy and ranking metrics, it performs significantly better than classic BPR that uses both types of implicit feedback in a naive way.

BPR^{++}. In [66], a graded preference relation scheme is introduced to extend the set of triples D_S used for training the model in the gradient descent phase. This new set D^{++}_S includes additional triples based on the graded observed feedback. Similar to the BPRC and ABPR approaches discussed before, the goal of the BPR^{++} technique is to adapt BPR to non-binary feedback, e.g., the confidence in the interaction, the number of times an interaction was observed, the recency of the interaction or the type of an interaction. The enlarged training set is defined as follows:

$$D^{++}_S := \{(u, i, j) | pweight(u, i) > pweight(u, j), i \in I, j \in I\} \tag{14.23}$$

The function $pweight(u, i)$ is a real-valued preference weight function that models the strength of the interaction between user u and item i, e.g., confidence,

time or rating. If there is no interaction between u and i, the preference weight is 0. Compared to the original set of training triples as shown before in Equation 14.2, the extended set D_S^{++} contains all triples of D_S. In addition, triples that would have been ignored in BPR because both items had observed feedback can now appear in D_S^{++} if they have a different preference weight.

The number of these additional triples $D_S^{++} \setminus D_S$ is however comparably small which means that these triples will not be often considered when using the random sampling strategy of BPR. The authors therefore introduce a weighted sampling approach, similar to the one presented in [46], which increases the sampling probability for the new triples in $D_S^{++} \setminus D_S$.

The BPR^{++} approach was benchmarked against the original BPR technique on two e-commerce datasets with implicit feedback and a MovieLens dataset with ratings. Different preference weight functions where used in the experiments to model the strength of the relevance including the interaction time, the number of interactions and – for the MovieLens data – the ratings. The results show that on the implicit feedback datasets the use of the interaction time with BPR^{++} significantly increased the accuracy (Precision@10 and Recall@10) when compared to BPR. On the MovieLens dataset this is also true when the preference strength is modeled by ratings. The number of interactions, however, did not have an impact on the recommendation accuracy.

Besides being able to improve the prediction accuracy, the adapted sampling strategy helps reducing the time needed for the gradient descent procedure to converge.

14.7 Summary and Outlook

Historically, fueled by competitions and publicly available datasets, research on recommender systems was largely focused on applications that are based on explicit user feedback. Only in recent years approaches based on implicit feedback have received more attention due to their high practical relevance. This chapter provided an overview on the use of implicit feedback in recommender systems. We presented examples of typical application scenarios and extended an established categorization of observable behavior for new domains that emerged with the Social Web and ubiquitous systems. The chapter furthermore reviewed state-of-the-art algorithmic approaches for implicit feedback as well as typical challenges that arise when dealing with implicit preference signals.

As an outlook on future developments all three key challenges identified by [58] particularly apply to recommendation systems operating on implicit user feedback: *scalability*, *better exploitation of user-contributed content* and *research infrastructure*, meaning more efficient mechanisms to evaluate the suitability of research contributions for actual live systems.

In our view, in particular the third point of assessing the practical value contribution of theoretical concepts and algorithms for the interaction with real

users creates the most diverse opportunities for future research. A recommender system based on explicit feedback requires the user to actively input feedback to a system in order to receive recommendations in return. In an implicit feedback recommender the user is monitored by a system that might therefore even proactively provide recommendations based on its observation of the user behavior and assumptions about the user's next goals. Thus, not only aspects of user experience but also the perceived intrusiveness receives particular importance in the context of exploiting implicit user feedback in real-world systems.

In general, the issue of the internal validity and reproducibility of research results is of particular relevance for research contributions on implicit user feedback. In [106] serious doubts about the internal validity of published results on explicit feedback datasets and their reproducibility with different algorithm libraries have been raised. Moreover, works on implicit feedback datasets often use proprietary data and commonly available benchmark datasets and algorithm libraries are still missing. Thus, we conclude this chapter with the hope that it will help to stimulate further progress on these lines.

References

1. Abel, F., Gao, Q., Houben, G.J., Tao, K.: Semantic enrichment of Twitter posts for user profile construction on the social web. In: Proceedings of the 8th Extended Semantic Web Conference on The Semanic Web: Research and Applications. ESWC '11, vol. Part II, pp. 375–389 (2011)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. pp. 207–216. SIGMOD '93, ACM (1993)
3. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering. pp. 3–14. ICDE '95 (1995)
4. Ali, K., van Stam, W.: TiVo: Making show recommendations using a distributed collaborative filtering architecture. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 394–401. KDD '04 (2004)
5. Armentano, M.G., Godoy, D., Amandi, A.: Topology-based recommendation of users in micro-blogging communities. Journal of Computer Science and Technology 27(3), 624–634 (2012)
6. Armstrong, R., Freitag, D., Joachims, T., Mitchell, T.: WebWatcher: A learning apprentice for the world wide web. In: AAAI Technical Report SS-95-08. pp. 6–12 (1995)
7. Ben-David, S., Lindenbaum, M.: Learning distributions by their density levels: A paradigm for learning without a teacher. Journal of Computer and System Sciences (1997)
8. Bogers, T.: Tag-based recommendation. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
9. Bohnert, F., Zukerman, I.: Personalised viewing-time prediction in museums. User Modeling and User-Adapted Interaction 24(4), 263–314 (2014)
10. Bonnin, G., Jannach, D.: Evaluating the quality of playlists based on hand-crafted samples. In: Proceedings of the 2013 International Society for Music Information Retrieval. pp. 263–268. ISMIR '13 (2013)
11. Bonnin, G., Jannach, D.: Automated generation of music playlists: Survey and experiments. ACM Computing Surveys 47(2), 26:1–26:35 (2014)
12. Bothorel, C., Lathia, N., Picot-Clemente, R., Noulas, A.: Location recommendation with social media data. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)

13. Budzik, J., Hammond, K.: Watson: Anticipating and contextualizing information needs. In: 62nd Annual Meeting Of The American Society For Information Science. pp. 727–740 (1999)
14. Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'El, N., Ronen, I., Uziel, E., Yogev, S., Chernov, S.: Personalized social search based on the user's social network. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. pp. 1227–1236. CIKM '09 (2009)
15. Castagnos, S., Jones, N., Pu, P.: Eye-tracking product recommenders' usage. In: Proceedings of the 2010 ACM Conference on Recommender Systems. pp. 29–36. RecSys '10 (2010)
16. Chau, P.Y.K., Ho, S.Y., Ho, K.K.W., Yao, Y.: Examining the effects of malfunctioning personalized services on online users' distrust and behaviors. Decision Support Systems 56, 180–191 (2013)
17. Chen, W.Y., Chu, J.C., Luan, J., Bai, H., Wang, Y., Chang, E.Y.: Collaborative filtering for orkut communities: Discovery of user latent behavior. In: Proceedings of the 18th International Conference on World Wide Web. pp. 681–690. WWW '09 (2009)
18. Cheng, Z., Shen, J.: Just-for-me: An adaptive personalization system for location-aware social music recommendation. In: Proceedings of International Conference on Multimedia Retrieval. pp. 185:185–185:192. ICMR '14 (2014)
19. Claypool, M., Le, P., Wased, M., Brown, D.: Implicit interest indicators. In: Proceedings of the 6th International Conference on Intelligent User Interfaces. pp. 33–40. IUI '01 (2001)
20. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the 2010 ACM Conference on Recommender Systems. pp. 39–46. RecSys '10 (2010)
21. Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B., Sampath, D.: The YouTube video recommendation system. In: Proceedings of the 2010 ACM Conference on Recommender Systems. pp. 293–296. RecSys '10 (2010)
22. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning. pp. 233–240. ICML '06 (2006)
23. De Pessemier, T., Dooms, S., Martens, L.: Context-aware recommendations through context and activity recognition in a mobile environment. Multimedia Tools and Applications 72(3), 2925–2948 (2014)
24. Denis, F.: PAC learning from positive statistical queries. In: Algorithmic Learning Theory, Lecture Notes in Computer Science, vol. 1501, pp. 112–126. Springer (1998)
25. Du, L., Li, X., Shen, Y.D.: User graph regularized pairwise matrix factorization for item recommendation. In: Tang, J., King, I., Chen, L., Wang, J. (eds.) Advanced Data Mining and Applications, Lecture Notes in Computer Science, vol. 7121, pp. 372–385. Springer (2011)
26. Gadanho, S.C., Lhuillier, N.: Addressing uncertainty in implicit preferences. In: Proceedings of the 2007 ACM Conference on Recommender Systems. pp. 97–104. RecSys '07 (2007)
27. Garcia, E., Romero, C., Ventura, S., Castro, C.: An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. User Modeling and User-Adapted Interaction 19(1-2), 99–132 (2009)
28. Gedikli, F., Jannach, D.: Improving recommendation accuracy based on item-specific tag preferences. ACM Transactions on Intelligent Systems and Technology 4(1), 11:1–11:19 (2013)
29. Gil, M., Pelechano, V.: Exploiting user feedback for adapting mobile interaction obtrusiveness. In: Proceedings of the 6th International Conference on Ubiquitous Computing and Ambient Intelligence. pp. 274–281. UCAmI '12 (2012)
30. Glazyrin, N.: Music recommendation system for million song dataset challenge. CoRR abs/1209.3286 (2012)

31. Guy, I.: People recommendation in social systems. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
32. Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S., Ofek-Koifman, S.: Personalized recommendation of social software items based on social relations. In: Proceedings of the 2009 ACM Conference on Recommender Systems. pp. 53–60. RecSys '09 (2009)
33. Guy, I., Zwerdling, N., Ronen, I., Carmel, D., Uziel, E.: Social media recommendation based on people and tags. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 194–201. SIGIR '10 (2010)
34. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. pp. 1–12. SIGMOD '00 (2000)
35. Han, S., He, D.: Network-based social search. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
36. Hariri, N., Mobasher, B., Burke, R.: Context-aware music recommendation based on latent topic sequential patterns. In: Proceedings of the 2012 ACM Conference on Recommender Systems. pp. 131–138. RecSys '12 (2012)
37. Hensley, C.B.: Selective dissemination of information (SDI). In: Proceedings of the 1963 Spring Joint Computer Conference. pp. 257–262. AFIPS '63 (1963)
38. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22(1), 5–53 (Jan 2004)
39. Hill, W.C., Hollan, J.D., Wroblewski, D., McCandless, T.: Edit wear and read wear. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 3–9. CHI '92 (1992)
40. Höök, K., Benyon, D., Munro, A.J.: Designing Information Spaces: The Social Navigation Approach. Springer Science & Business Media (2012)
41. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. pp. 263–272. ICDM '08 (2008)
42. Jannach, D., Hegelich, K.: A case study on the effectiveness of recommendations in the mobile internet. In: Proceedings of the 2009 ACM Conference on Recommender Systems. pp. 205–208. RecSys '09 (2009)
43. Jannach, D., Karakaya, Z., Gedikli, F.: Accuracy improvements for multi-criteria recommender systems. In: Proceedings of the 13th ACM Conference on Electronic Commerce. pp. 674–689. EC '12 (2012)
44. Jannach, D., Lerche, L., Jugovac, M.: Adaptation and evaluation of recommendations for short-term shopping goals. In: Proceedings of the 2015 ACM Conference on Recommender Systems. pp. 211–218. RecSys '15, ACM (2015)
45. Jannach, D., Lerche, L., Kamehkhosh, I.: Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In: Proceedings of the 2015 ACM Conference on Recommender Systems. pp. 187–194. RecSys '15, ACM (2015)
46. Jannach, D., Lerche, L., Kamehkhosh, I., Jugovac, M.: What recommenders recommend: An analysis of recommendation biases and possible countermeasures. User Modeling and User-Adapted Interaction 25(5), 427–491 (2015)
47. Jannach, D., Zanker, M., Ge, M., Gröning, M.: 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. EC-WEB '12 (2012)
48. Jawaheer, G., Szomszor, M., Kostkova, P.: Comparison of implicit and explicit feedback from an online music recommendation service. In: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems. pp. 47–51. HetRec '10 (2010)

49. Jawaheer, G., Weller, P., Kostkova, P.: Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. ACM Transactions on Interactive Intelligent Systems 4(2), 8:1–8:26 (2014)
50. Kanagal, B., Ahmed, A., Pandey, S., Josifovski, V., Yuan, J., Garcia-Pueyo, L.: Supercharging recommender systems using taxonomies for learning user purchase behavior. Proceedings of the VLDB Endowment 5(10), 956–967 (Jun 2012)
51. Ke, T., Yang, B., Zhen, L., Tan, J., Li, Y., Jing, L.: Building high-performance classifiers using positive and unlabeled examples for text classification. In: Advances in Neural Networks (ISNN '12), Lecture Notes in Computer Science, vol. 7368, pp. 187–195. Springer (2012)
52. Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: A bibliography. SIGIR Forum 37(2), 18–28 (2003)
53. Khalili, A., Wu, C., Aghajan, H.: Autonomous learning of user's preference of music and light services in smart home applications. In: Behavior Monitoring and Interpretation Workshop at German AI Conference (2009)
54. Kliegr, T., Kuchar, J.: Orwellian Eye: Video recommendation with Microsoft Kinect. In: Proceedings 21st European Conference on Artificial Intelligence. pp. 1227–1228. ECAI/PAIS '14 (2014)
55. Kluver, D., Ekstrand, M., Konstan, J.: Collaborative filtering. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
56. Kobsa, A., Koenemann, J., Pohl, W.: Personalised hypermedia presentation techniques for improving online customer relationships. Knowledge Engineering Review 16(2), 111–155 (2001)
57. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: Applying collaborative filtering to usenet news. Communications of the ACM 40(3), 77–87 (1997)
58. Konstan, J.A., Riedl, J.: Recommender systems: From algorithms to user experience. User Modeling and User-Adapted Interaction 22(1-2), 101–123 (2012)
59. Kordumova, S., Kostadinovska, I., Barbieri, M., Pronk, V., Korst, J.: Personalized implicit learning in a music recommender system. In: User Modeling, Adaptation, and Personalization, Lecture Notes in Computer Science, vol. 6075, pp. 351–362. Springer (2010)
60. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 426–434. KDD '08 (2008)
61. Koren, Y.: Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD) 4(1), 1:1–1:24 (2010)
62. Krohn-Grimberghe, A., Drumond, L., Freudenthaler, C., Schmidt-Thieme, L.: Multi-relational matrix factorization using bayesian personalized ranking for social network data. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. pp. 173–182. WSDM '12 (2012)
63. Lee, D., Brusilovsky, P.: Social link-based recommendations. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
64. Lee, D., Park, S., Kahng, M., Lee, S., Lee, S.g.: Exploiting contextual information from event logs for personalized recommendation. In: Computer and Information Science 2010, Studies in Computational Intelligence, vol. 317, pp. 121–139. Springer (2010)
65. Lee, T.Q., Park, Y., Park, Y.T.: A time-based approach to effective recommender systems using implicit feedback. Expert Systems with Applications 34(4), 3055–3062 (May 2008)
66. Lerche, L., Jannach, D.: Using graded implicit feedback for bayesian personalized ranking. In: Proceedings of the 2014 ACM Conference on Recommender Systems. pp. 353–356. RecSys '14 (2014)
67. Lerchenmüller, B., Wörndl, W.: Inference of user context from GPS logs for proactive recommender systems. In: AAAI Workshop Activity Context Representation: Techniques and Languages, AAAI Technical Report WS-12-05 (2012)

68. Li, H., Wang, Y., Zhang, D., Zhang, M., Chang, E.Y.: PFP: Parallel FP-growth for query recommendation. In: Proceedings of the 2008 ACM Conference on Recommender Systems. pp. 107–114. RecSys '08 (2008)
69. Li, X.L., Liu, B.: Learning from positive and unlabeled examples with different data distributions. In: Machine Learning: ECML '05, Lecture Notes in Computer Science, vol. 3720, pp. 218–229. Springer (2005)
70. Lieberman, H.: Letizia: An agent that assists web browsing. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence. IJCAI '95, vol. 1, pp. 924–929 (1995)
71. Lin, J., Sugiyama, K., Kan, M.Y., Chua, T.S.: Addressing cold-start in app recommendation: Latent user models constructed from Twitter followers. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 283–292. SIGIR '13 (2013)
72. Lin, K.H., Chung, K.H., Lin, K.S., Chen, J.S.: Face recognition-aided IPTV group recommender with consideration of serendipity. International Journal of Future Computer and Communication 3(2), 141–147 (2014)
73. Lin, W., Alvarez, S.A., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining and Knowledge Discovery 6(1), 83–105 (2002)
74. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (Jan 2003)
75. Liu, N.N., Xiang, E.W., Zhao, M., Yang, Q.: Unifying explicit and implicit feedback for collaborative filtering. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. pp. 1445–1448. CIKM '10 (2010)
76. Liu, Y., Zhao, P., Sun, A., Miao, C.: AdaBPR: A boosting algorithm for item recommendation with implicit feedback. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence. IJCAI '15 (2015)
77. Marlin, B.M., Zemel, R.S., Roweis, S., Slaney, M.: Collaborative filtering and the missing at random assumption. In: Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence. pp. 267–275. UAI '07 (2007)
78. McFee, B., Bertin-Mahieux, T., Ellis, D.P., Lanckriet, G.R.: The million song dataset challenge. In: Proceedings of the 21st International Conference Companion on World Wide Web. pp. 909–916. WWW '12 Companion (2012)
79. Mladenic, D.: Personal WebWatcher: Design and implementation, J. Stefan Institute, Slovenia, Technical report IJS-DP-7472 (1996)
80. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on web usage mining. Communications of the ACM 43(8), 142–151 (2000)
81. Moore, J.L., Chen, S., Joachims, T., Turnbull, D.: Learning to embed songs and tags for playlist prediction. In: Proceedings of the 2012 International Society for Music Information Retrieval Conference. pp. 349–354. ISMIR '12 (2012)
82. Morita, M., Shinoda, Y.: Information filtering based on user behavior analysis and best match text retrieval. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 272–281. SIGIR '94 (1994)
83. Nichols, D.M.: Implicit rating and filtering. In: Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering (1997)
84. Oard, D.W.: Modeling information content using observable behavior. In: Proceedings of the 64th Annual Conference of the American Society for Information Science and Technology (2001)
85. Oard, D.W., Kim, J.: Implicit feedback for recommender systems. In: Proceedings of the 1998 AAAI Workshop on Recommender Systems (1998)
86. O'Mahoney, M., Smyth, B.: From opinions to recommendations. In: Brusilovsky, P., He, D. (eds.) Social Information Access, LNCS, vol. 10100, p. in this volume. Springer, Heidelberg (2017)
87. Palanivel, K., Sivakumar, R.: A study on implicit feedback in multicriteria e-commerce recommender system. Journal of Electronic Commerce Research 11(2), 140–156 (2010)

88. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R., Scholz, M., Yang, Q.: One-class collaborative filtering. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. pp. 502–511. ICDM '08 (2008)
89. Pan, W., Chen, L.: CoFiSet: Collaborative filtering via learning pairwise preferences over item-sets. In: Proceedings of the 2013 SIAM International Conference on Data Mining. pp. 180–188 (2013)
90. Pan, W., Chen, L.: GBPR: Group preference based bayesian personalized ranking for one-class collaborative filtering. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence. pp. 2691–2697. IJCAI '13 (2013)
91. Pan, W., Zhong, H., Xu, C., Ming, Z.: Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. Knowledge-Based Systems 73(0), 173–180 (2015)
92. Paquet, U., Koenigstein, N.: One-class collaborative filtering with random graphs. In: Proceedings of the 22nd International Conference on World Wide Web. pp. 999–1008. WWW '13 (2013)
93. Parra, D., Amatriain, X.: Walk the talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. In: Proceedings of the 19th International Conference on User Modeling, Adaptation, and Personalization. pp. 255–268. UMAP '11 (2011)
94. Parra, D., Karatzoglou, A., Yavuz, I., Amatriain, X.: Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. In: Proceedings of the CARS '11 (2011)
95. Partridge, K., Price, B.: Enhancing mobile recommender systems with activity inference. In: User Modeling, Adaptation, and Personalization, Lecture Notes in Computer Science, vol. 5535, pp. 307–318. Springer (2009)
96. Pennacchiotti, M., Gurumurthy, S.: Investigating topic models for social media user recommendation. In: Proceedings of the 20th International Conference Companion on World Wide Web. pp. 101–102. WWW '11 (2011)
97. Pilászy, I., Zibriczky, D., Tikk, D.: Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In: Proceedings of the 2010 ACM Conference on Recommender Systems. pp. 71–78. RecSys '10 (2010)
98. Pitman, A., Zanker, M.: An empirical study of extracting multidimensional sequential rules for personalization and recommendation in online commerce. In: 10. Internationale Tagung Wirtschaftinformatik. pp. 180–189 (2011)
99. Pizzato, L., Chung, T., Rej, T., Koprinska, I., Yecef, K., Kay, J.: Learning user preferences in online dating. In: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. ECML-PKDD, Preference Learning Workshop (2010)
100. Rendle, S.: Factorization machines with libFM. ACM Transactions on Intelligent Systems and Technology 3(3), 57:1–57:22 (May 2012)
101. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 452–461. UAI '09 (2009)
102. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web. pp. 811–820. WWW '10, ACM (2010)
103. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining. pp. 81–90. WSDM '10 (2010)
104. Rodden, K., Fu, X., Aula, A., Spiro, I.: Eye-mouse coordination patterns on web search results pages. In: Extended Abstracts on Human Factors in Computing Systems. pp. 2997–3002. CHI EA '08 (2008)
105. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y., Merom, R.: Suggesting friends using the implicit social graph. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 233–242. KDD '10, ACM (2010)

106. Said, A., Bellogín, A.: Comparative recommender system evaluation: Benchmarking recommendation frameworks. In: Proceedings of the 2014 ACM Conference on Recommender Systems. pp. 129–136. RecSys '14 (2014)
107. Sakagami, H., Kamba, T.: Learning personal preferences on online newspaper articles from user behaviors. Computer Networks and ISDN Systems 29(8-13), 1447–1455 (1997)
108. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: Proceedings of the 2nd ACM Conference on Electronic Commerce. pp. 158–167. EC '00 (2000)
109. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471 (Jul 2001)
110. Sen, S., Vig, J., Riedl, J.: Tagommenders: Connecting users to items through tags. In: Proceedings of the 18th International World Wide Web Conference. pp. 671–680. WWW '09 (2009)
111. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. Journal of Machine Learning Research 6, 1265–1295 (2005)
112. Shapira, B., Taieb-Maimon, M., Moskowitz, A.: Study of the usefulness of known and new implicit indicators and their optimal combination for accurate inference of users interests. In: Proceedings of the 2006 ACM Symposium on Applied Computing. pp. 1118–1119. SAC '06 (2006)
113. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A.: xCLiMF: Optimizing expected reciprocal rank for data with multiple levels of relevance. In: Proceedings of the 2013 ACM Conference on Recommender Systems. pp. 431–434. RecSys '13 (2013)
114. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: CLiMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the 2012 ACM Conference on Recommender Systems. pp. 139–146. RecSys '12 (2012)
115. Sohn, T., Li, K.A., Griswold, W.G., Hollan, J.D.: A diary study of mobile information needs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 433–442. CHI '08 (2008)
116. Srebro, N., Jaakkola, T., et al.: Weighted low-rank approximations. In: Proceedings of the Twentieth International Conference on Machine Learning. pp. 720–727. ICML '03 (2003)
117. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in Twitter to improve information filtering. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 841–842. SIGIR '10 (2010)
118. Tapia, E.M., Intille, S.S., Larson, K.: Activity recognition in the home using simple and ubiquitous sensors. In: Pervasive Computing, Lecture Notes in Computer Science, vol. 3001, pp. 158–175. Springer (2004)
119. Tavakol, M., Brefeld, U.: Factored MDPs for detecting topics of user sessions. In: Proceedings of the 2014 ACM Conference on Recommender Systems. pp. 33–40. RecSys '14 (2014)
120. Véras, D., Prota, T., Bispo, A., Prudêncio, R., Ferraz, C.: A literature review of recommender systems in the television domain. Expert Systems with Applications 42(22), 9046–9076 (2015)
121. Wang, J., Zhu, J.: On statistical analysis and optimization of information retrieval effectiveness metrics. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 226–233. SIGIR '10 (2010)
122. Wang, S., Zhou, X., Wang, Z., Zhang, M.: Please spread: Recommending Tweets for retweeting with implicit feedback. In: Proceedings of the 2012 Workshop on Data-driven User Behavioral Modelling and Mining from Social Media. pp. 19–22. DUBMMSM '12 (2012)
123. Ward, G., Hastie, T., Barry, S., Elith, J., Leathwick, J.R.: Presence-only data and the EM algorithm. Biometrics 65(2), 554–563 (2009)

124. White, R.W., Ruthven, I., Jose, J.M.: A study of factors affecting the utility of implicit relevance feedback. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 35–42 (2005)
125. Yajima, Y.: One-class support vector machines for recommendation tasks. In: Proceedings of the 10th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. pp. 230–239. PAKDD '06 (2006)
126. Yu, H., Han, J., Chang, K.C.C.: PEBL: Positive example based learning for web page classification using SVM. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 239–248. KDD '02 (2002)
127. Yu, Z., Zhou, X.: TV3P: An adaptive assistant for personalized TV. IEEE Transactions on Consumer Electronics 50(1), 393–399 (2004)
128. Zanker, M., Jessenitschnig, M.: Collaborative feature-combination recommender exploiting explicit and implicit user feedback. In: Proceedings of the IEEE Conference on Commerce and Enterprise Computing. pp. 49–56. CEC '09 (7 2009)
129. Zhang, Y., Callan, J.: Combining multiple forms of evidence while filtering. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. pp. 587–595. HLT '05 (2005)
130. Zhang, Y.C., Séaghdha, D.O., Quercia, D., Jambor, T.: Auralist: Introducing serendipity into music recommendation. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. pp. 13–22. WSDM '12 (2012)
131. Zhao, S., Du, N., Nauerz, A., Zhang, X., Yuan, Q., Fu, R.: Improved recommendation based on collaborative tagging behaviors. In: Proceedings of the 13th International Conference on Intelligent User Interfaces. pp. 413–416. IUI '08 (2008)
132. Zheng, V.W., Zheng, Y., Xie, X., Yang, Q.: Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. Artificial Intelligence 184/185, 17–37 (2012)
133. Zibriczky, D., Hidasi, B., Petres, Z., Tikk, D.: Personalized recommendation of linear content on interactive TV platforms: Beating the cold start and noisy implicit user feedback. In: UMAP '12 Workshop on TV and Multimedia Personalization. UMAP '12 (2012)
134. Zigoris, P., Zhang, Y.: Bayesian adaptive user profiling with explicit & implicit feedback. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management. pp. 397–404. CIKM '06 (2006)