Chapter 11
Benchmarking

A Methodology for Ensuring the Relative Quality of Recommendation Systems in Software Engineering

Alan Said, Domonkos Tikk, and Paolo Cremonesi

Abstract This chapter describes the concepts involved in the process of benchmarking of recommendation systems. Benchmarking of recommendation systems is used to ensure the quality of a research system or production system in comparison to other systems, whether algorithmically, infrastructurally, or according to any sought-after quality. Specifically, the chapter presents evaluation of recommendation systems according to recommendation accuracy, technical constraints, and business values in the context of a multi-dimensional benchmarking and evaluation model encompassing any number of qualities into a final comparable metric. The focus is put on quality measures related to recommendation accuracy, technical factors, and business values. The chapter first introduces concepts related to evaluation and benchmarking of recommendation systems, continues with an overview of the current state of the art, then presents the multi-dimensional approach in detail. The chapter concludes with a brief discussion of the introduced concepts and a summary.

A. Said (✉)
Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
e-mail: alan@cwi.nl

D. Tikk
Gravity R&D, Budapest, Hungary

Óbuda University, Budapest, Hungary
e-mail: domonkos.tikk@gravityrd.com; domonkos.tikk@nik.uni-obuda.hu

P. Cremonesi
Politecnico di Milano, Milano, Italy
e-mail: paolo.cremonesi@polimi.it

M.P. Robillard et al. (eds.), *Recommendation Systems in Software Engineering*,
DOI 10.1007/978-3-642-45135-5_11, © Springer-Verlag Berlin Heidelberg 2014

11.1 Introduction

Benchmarking is a structural approach to quality engineering and management [7]; essentially it is a comparison process aimed at finding the best practice for a given well-specified problem. The concept of benchmarking originated in optimizing business processes by investigating and analyzing industry standards, comparing them to the one applied in the investigator's own organization, and creating an implementation plan with predefined goals and objectives to improve the quality and performance of the evaluated process. In the last few decades, benchmarking has also become very popular in scientific research and software engineering, driven by the need to identify best-in-class approaches or algorithms for scientific problems, and to facilitate various stages of the software development lifecycle, including automated code-testing [18].

The process of traditional benchmarking is built up from the following steps: (1) design and target specification, (2) data collection, (3) evaluation and analysis, and (4) implementation of improvements. Scientific benchmarking, on the other hand, mainly focuses on providing a means for comparison and exploration of novel ideas on a dataset collected for the given purpose,[1] and puts less emphasis on the implementation of the improvements in an industrial environment.

Due to their origins in the research community, recommendation systems are primarily evaluated using accuracy-oriented metrics, such as precision, recall, root-mean-squared error, etc. [22]. As a typical example, we refer to the Netflix Prize competition [31] (see more details in Sect. 11.2). However, these measures only represent one type of performance, namely the objective recommendation accuracy, not taking into consideration software engineering and business aspects, technical constraints, and subjective user-centric values, thus creating an unbalanced focus on only one dimension of the evaluation spectrum. This lack of balance makes benchmarking of recommendation systems in different domains and settings difficult, if not impossible. The business and technical constraints are largely neglected in traditional algorithmic evaluation [22]. Even in cases where multi-objective evaluation is applied, the evaluation often focuses only on recommendation accuracy [e.g., 24]. The broadly accepted philosophy is that the higher the accuracy (or lower the error) metrics are, the better the recommendation system performs [22]. In order to objectively estimate the utility of a recommendation system, from all perspectives, the complete spectrum of recommendation quality should be evaluated, especially in contexts where accuracy is not necessarily the ultimate goal.

In real-world scenarios, business- and technology-centered measures are just as important, if not more so, than accuracy alone. An evaluation model incorporating all three values was presented by Said et al. [37]. This model, if applied in the context of a real-world, market-driven recommendation system, simplifies the

[1] See for example the UCI Machine Learning Repository that contains a large selection of machine learning benchmark datasets. Recommendation-system-related benchmark datasets can also be found in KONECT, e.g., under category ratings.

algorithm-to-algorithm comparison of recommendation systems. Especially from a software engineering perspective, evaluating technical constraints is particularly important in order to create a well-functioning system.

Throughout this chapter, we use the evaluation model outlined by Said et al. [37] to discuss and reason whether the most important challenges are related to large-scale, real-time, business-driven, or highly accurate recommendations. The context of the recommendation is often related to the setting in which the recommendation is to be presented, and to what quality is important in the specific setting. For instance, below we present a few examples where recommendation systems can be (and often are) deployed, and, given the diversity of the services, where the sought-after qualities in each service need not to be the same.

Video/Music-on-Demand. Video-on-demand (VOD) and music-on-demand (MOD) are services where multimedia content is provided to the users on request; examples of these include Netflix and Spotify. The difference from other media, e.g., radio or live TV, is the user-driven availability of the content. This instant availability of content creates a certain context in which the recommendations are most often consumed directly and not stored for later viewing, listening, etc. Additionally, there is a business context in which a specific item might be a preferred recommendation from the provider based on infrastructure, revenue, or other factors. It should however still represent a suitable recommendation from the user's perspective.

Linear TV. In linear (or traditional) television (TV) where the delivery of content is driven by one provider to a large audience without any personalization, the selection of items is limited and quickly changing. In this context, it is imperative for the recommendation system to adapt to the currently available items. However, the user- and business-focused aspects should not be overlooked as the utility of quickly updating but poorly performing recommendation systems is low both for the user and for the service operator.

Webshop. In a webshop setting, the user- and business-focused aspects of recommendations might be different, since users seek quality products at low prices, while the business is focusing on maximizing the revenue/profit per user visit. The latter could potentially be achieved by recommending quality and more expensive/larger margin products. Therefore the utility of the recommender is different for the two aspects, which may necessitate the implementation of a recommendation algorithm that trades off between the different goals.

News Portal. On a news portal, users seek interesting content to read, and their "currency" is the time they spend on the site, visit frequency, and the number of pages visited. Since users' browsing time is usually limited per visit, the quality of recommendations, and therefore the user satisfaction and loyalty is often mirrored in increased visit frequency [28]. On the service provider's end, the business goal is to increase the total number of page views, since the ad display-based business model scales with page views. Alternatively, pay-per view or subscriber content could

provide an additional revenue stream; in such a case the recommender algorithm should also identify those users that are willing to pay for the content. An additional user-focused quality measure is the diversity of the recommendation that also requires good adaptability from the recommendation system to capture, in real-time, the user's actual interest.

Internet Marketplaces. On Internet marketplaces, such as auction sites (e.g., eBay) or classified media sites (e.g., leboncoin, craigslist), the perceived quality of recommendations from the user depends on how quickly the algorithm can adapt to the actual need of the visit. On the business side, it is important to keep users engaged and active in using the service while being able to sell value-added services for advertisers/sellers. Therefore the recommendation algorithm should again trade off between user satisfaction and the site's business goals.

The requirements on the recommendation algorithms deployed in each example are clearly different; the implication that follows is that they should also be evaluated differently. The multidimensional evaluation model presented in this chapter allows for this, while still keeping a reasonable means of comparison.

11.2 Benchmarking and Evaluation Settings

We explain the process of benchmarking based on the Netflix Prize (NP) example, which is by far the most widely known benchmarking event and dataset for recommendation systems. Recall that the benchmarking process has the following steps: (1) design and target specification, (2) data collection, (3) evaluation and analysis, and (4) implementation of improvements.

Netflix initiated the contest in order to improve their in-house recommendation system—called Cinematch—that provides movie recommendations to their customers. Although the ultimate goal of Netflix was to improve or replace Cinematch with a recommendation system that would provide more satisfactory recommendations to the end-users and thus improve their business,[2] they selected a less sensitive and essentially simpler task as a proxy to benchmark algorithms of the participants.

At Netflix, users can express their movie preferences by rating movies on a 1–5 scale. The aim of the competition was to improve the prediction accuracy of user ratings, that is, participants in the competition had to create algorithms to predict a set of unreported user ratings (called the Qualifying set), using a set of reported user ratings (called the Training set). Netflix released a large rating dataset (for comparison to other datasets, see Table 11.1) as follows [43] (see Fig. 11.1 for an overview). Netflix selected a random subset of users from their entire customer base

[2]Better recommendations postpone or eliminate the *content glut effect* [32]—a variation on the idea of information overload—and thus increases customer lifetime, which is translated into additional revenue of Netflix's monthly plan based subscription service.

Table 11.1 Benchmark datasets for recommendation tasks. Starred datasets contain implicit ratings; density is given as a percentage

Name	Domain	Events	Users	Items	Density
Jester	jokes	4,136,360	73,421	100	56.34
Book-crossing	book	1,149,780	278,858	271,379	0.001
MovieLens 100k	movie	100,000	943	1682	6.30
MovieLens 1M	movie	1,000,000	6040	3900	4.25
MovieLens 10M	movie	10,000,000	71,567	10,681	1.31
Netflix	movie	100,480,507	480,189	17,7	1.17
CAMRa2010 (Moviepilot)	movie	4,544,409	105,137	25,058	0.002
CAMRa2011 (Moviepilot)	movie	4,391,822	171,67	29,974	0.001
CAMRa2010 (Filmtipset time)	movie	5,862,464	34,857	53,6	0.003
CAMRa2010 (Filmtipset social)	movie	3,075,346	16,473	24,222	0.008
Last.fm 1K*	music	19,150,868	992	176,948	10.91
Last.fm 360K*	music	17,559,530	359,347	294,015	0.016
Yahoo Music (KDD Cup 2011)	music	262,810,175	1,000,990	624,961	0.042
Mendeley*	publications	4,848,724	50	3,652,285	0.002
LibimSeTi	dating	17,359,346	135,359	168,791	0.076
Delicious	tags	420,000,000	950	132,000,000	$3 \cdot 10^{-6}$
Koders-log-2007	code search	5M + 5M	3,187,969	see note 5	see note 5

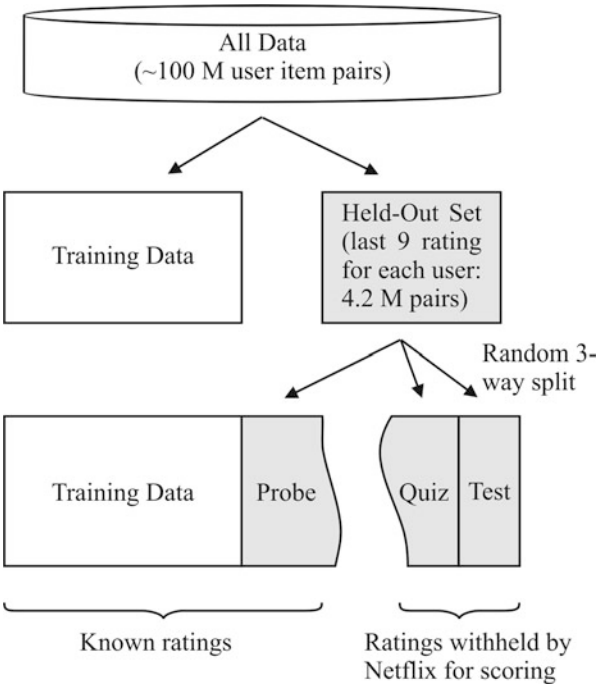

Fig. 11.1 The benchmark dataset of the Netflix Prize [adapted from 6]

with at least 20 ratings in a given period. A *Hold-Out set* was created from the 9 most recent ratings of each of these users,[3] consisting of about 4.2 million ratings. The remaining data formed the Training set. The ratings of the Hold-Out set were split randomly into three subsets of equal size: Quiz, Test, and Probe. The Probe set was released with ratings, primarily to allow the competitors to self-evaluate their algorithms, although they also used this data for training purposes in submitting the final predictions. The Quiz and Test sets formed the Qualifying set for which the actual ratings were withheld in order to evaluate competitors. The Quiz/Test split of the Qualifying set was unknown to the public. Netflix adopted root-mean-squared error (RMSE) as their evaluation measure to compare the algorithms. Participants had to submit only the predictions on the Qualifying set, not their algorithms. The organizers returned the RMSE of the submissions on the Quiz set, which is also reported on a public leaderboard. The RMSE on the Test set was withheld by Netflix, in order to retain some data for which competitors were unable to adjust their algorithms.

In 2009, the 1 million dollar prize was awarded to an ensemble algorithm, which successfully outperformed Cinematch by more than 10 % at RMSE. However, in the end the ensemble was not deployed by Netflix; one of the reasons behind this was simply that it would not scale-up to the amount of data available in the production environment [2]. The inability to deploy this ensemble algorithm should serve as a motivation as to why recommendation systems need to be evaluated in terms other than recommendation accuracy only. Because real-world recommendation systems are software tools, considering software engineering related parameters at their evaluation is essential.

Other benchmarking events, organized by academia and industry—e.g., KDD Cup 2011 [25], various installations of the ECML/PKDD Discovery Challenge, the Overstock RecLab Prize, etc.—all focused on one dimension of the recommendation quality, namely, recommendation accuracy. Even though a considerable amount of time has passed since the Netflix Prize competition, recommendation systems are still evaluated and benchmarked in a similar fashion. Recent examples of this include the 2013 Recommender Systems Challenge where again only RMSE was used for comparing the algorithms to one another.

In the remaining part of this section, aspects of traditional evaluation (e.g., accuracy) are presented, followed by a description on how these can be applied in the multi-dimensional benchmarking model also presented in this section. A more in-depth perspective of evaluation metrics is provided by Avazpour et al. [3] in Chap. 10.

[3]The date-based partition of the NP dataset into Training/Testing sets reflects the original aim of recommendation systems, which is the prediction of future interest of users from their past ratings/activities.

11.2.1 Datasets

Traditional evaluation of recommendation systems (based on accuracy metrics) requires a dataset on which the recommendation algorithms can be trained and validated. These range from user–item interaction datasets used for item recommendation, e.g., the Netflix Prize dataset, to more engineering-focused datasets containing API changes across versions of software or source code search queries and downloads, e.g., the Koders-log [4]. Table 11.1 shows some of the most common datasets used for these purposes together with their domains and sizes.[4]

Looking at attributes such as density, it is reasonable to believe that recommendation accuracy results on (for example) the Jester dataset are not directly comparable to (for example) the Delicious dataset. Similarly, comparing the scalability or speed of an algorithm using the Movielens 100k dataset and the KDD Cup 2011 dataset would not be fair either.[5]

11.2.2 Toolkits

When benchmarking recommendation algorithms, one of the aspects that affects factors such as scalability or even the accuracy of the system is the implementation of the algorithm itself. There exist several open source frameworks that are commonly used in research and industry; some of these are specialized on one or a few specific algorithms or recommendation contexts, whereas others provide very broad machine learning (ML) and natural language processing (NLP) libraries.

Table 11.2 shows some of the most common recommendation frameworks currently available together with their specific features.[6] Even though most of these frameworks have similar implementations of the most common algorithms (e.g., k-nearest neighbors [15]), due to the differences of the implementation languages, running time and memory usage may vary even if the same datasets, algorithms, and hardware are used.

11.2.3 Accuracy and Error Metrics

Traditional metrics measure concepts related to dataset-specific factors; often these are measures found in or based on similar concepts in statistics, radiology,

[4] Additional recommendation datasets can be found at the Recommender Systems Wiki.

[5] Due to the different context of this dataset, no number of items is given as the dataset instead contains two sets of event types (*search* and *download*). A density cannot be calculated as there is no fixed set of items.

[6] See mloss.org for additional general ML software and the Recommender Systems Wiki for recommendation-specific software.

Table 11.2 Common frameworks used for recommendation both in research and production systems

Name	License	Language	Type
CofiRank	MPL	C++	collaborative filtering
Crab	BSD	Python	recommendation
EasyRec	GPL v2	Java	recommendation
GraphLab	Apache 2.0	C++	high performance computation
Lenskit	LGPL v2.1	Java	recommendation
Mahout	Apache 2.0	Java	general ML
MyMediaLite	GPL	C# & Java	recommendation
PREA	BSD	Java	CF algorithms
Python-recsys	N/A	Python	recommendation
RapidMiner	AGPL	Java	ML, NLP & data mining
Recommendable	MIT	Ruby	recommendation
Recommender 101	Custom	Java	recommendation
Recommenderlab	GPL v2	R	recommendation
Svdfeature	Apache 2.0	C++	matrix factorization
Waffles	LGPL	C++	ML and data mining

medicine, etc. [21]. We overview a few such metrics here; for a more complete overview of recommendation evaluation measures, see Chap. 10 [3].

Classification Accuracy

Classification accuracy metrics measure to what extent a recommendation system is able to correctly classify items as interesting or not. Examples are precision and recall, which require the ratings to be mapped onto a binary relevance scale (relevant vs. not relevant). The number of recommendations returned by the recommender algorithm relates to precision and recall. Recall is typically used with a fixed number of recommended items (5–50); this setting reflects the online usage, when users receive a limited number of recommendations during a visit. Other classification accuracy metrics are the mean average precision [41], the receiver operating characteristic (ROC) curve, area under curve (AUC) [22], customer ROC (CROC) [40], etc.

Predictive Accuracy Metrics

Predictive accuracy metrics measure how a recommendation system can predict the ratings of users. Since rated items have an order, predictive accuracy metrics can also be used to measure a system's ability to rank items. The mean absolute error (MAE) and the root-mean-squared error (RMSE) are widely used metrics, and several variants of these exist. Several authors [9, 22] report that predictive accuracy metrics are not always appropriate: errors in the recommendation systems' predicted

ratings only affect the user when it results in erroneously classifying an interesting item as not interesting or vice versa.

Coverage Metrics

Coverage metrics measure the percentage of items for which the recommendation system can make predictions or recommendations [48]. A recommendation system cannot always generate a prediction since there might be insufficient data. There are two types of coverage identified by Herlocker et al. [22]: *prediction coverage*, the percentage of items in the input domain of the recommendation system for which it is able to make recommendations; and *catalog coverage*, the percentage of items in the output range of the recommendation system that it will ever present within a recommendation. A higher coverage means that the system is able to support decision making in more situations. Coverage cannot be considered independently from accuracy: a recommendation system can possibly achieve high coverage by making spurious predictions, but this has repercussions on accuracy.

Confidence Metrics

Confidence metrics measure how certain the recommendation system is about the accuracy of the recommendations. Extremely large or small predictions are often based on a small number of user ratings (i.e., high accuracy, low confidence). As the number of ratings grows, the prediction will usually converge to the mean (i.e., low accuracy, high confidence). Recommendation systems have different approaches to deal with confidence. Either they discard items with a confidence level that is below a certain threshold or display the confidence of a recommendation to the user. There is no general consensus on how to measure recommendation system confidence, since classical metrics based on statistical significance tests cannot be easily applied to all the algorithms [29].

Learning Rate Metrics

Many recommendation systems incorporate algorithms that gradually become better in recommending items. For instance, collaborative filtering (CF) algorithms are likely to perform better when more ratings are available. The learning rate measures the recommendation system's ability to cope with the cold start problem, i.e., how much historical data is needed before an algorithm can produce "good" recommendations. Three different types are overall learning rate, per item learning rate, and per user learning rate. Though the cold start problem is widely recognized by researchers, the evaluation of a recommendation system's learning rate has not yet been extensively covered in the literature and no specific metrics exist [39].

Diversity Metrics

Diversity of items in a recommendation list is an important factor for the usefulness of a recommendation. For instance, a user watching the first episode of the film "The Lord of the Rings" might receive recommendations for the sequel movies, which may be considered trivial. According to Ziegler et al. [48], diversity has a large effect on the usefulness of recommendation lists and therefore there is the need to define an intra-list similarity metric. The intra-list similarity metric is a measure for the diversity of a recommendation list.

Novelty and Serendipity Metrics

A recommendation system can produce highly accurate recommendations, have reasonably good coverage and diversity, and still not satisfy a user if the recommendations are trivial [44]. Novelty and serendipity are two closely related dimensions for non-obviousness [22]. Serendipity is the experience of discovering an unexpected and fortuitous item. This definition contains a notion of unexpectedness, i.e., the novelty dimension. Novelty and serendipity metrics thus measure the non-obviousness of recommendations and penalize "*blockbuster*" (i.e., common or popular) recommendations. The few existing suggestions on how to measure novelty and serendipity are limited to on-line analysis [14, 35].

User Satisfaction Metrics

In our context, *user satisfaction* is defined as the extent to which a user is supported in coping with the information overload problem.[7] This is a somewhat vague aspect and therefore it is difficult to measure [14,35]. All the previously defined metrics can support and/or inhibit user satisfaction to some extent. Studies that investigated user satisfaction with respect to recommendation systems are scarce, mainly because of the difficulties in performing on-line testing [35].

11.2.4 One-Dimensional Evaluation

Traditional recommendation system benchmarking commonly evaluates only one dimension of the recommender, i.e., quantitative recommendation accuracy. Moreover, even traditional recommendation accuracy driven evaluation may have

[7]Editors' note: More broadly, recommendation systems in software engineering do not only or always deal with the information overload problem [46]; thus, the definition of user satisfaction needs to be broadened in such situations.

Table 11.3 A user–item matrix divided into a training set (the top half) and a test set (the bottom half)

	u_1	u_2	u_3	u_4	u_5
i_1	1	1	0	0	1
i_2	1	0	1	1	1
i_3	0	0	0	1	0
i_4	1	0	1	0	1
i_5	0	0	1	1	0
i_6	0	0	0	1	0

drawbacks when applied together with an improper evaluation setting. To illustrate this we focus on the context of user-centric evaluation in a traditional user–item interaction scenario.

Consider this top-n recommendation example. We have a user-item interaction matrix, as shown in Table 11.3. The table shows a matrix of 5 users and 6 items and their interactions, where each one represents an interaction (rating, purchase, etc.), and each zero the lack thereof.

The training/test split is illustrated by the line in the middle of the table. In this case, a traditional evaluation approach will only recognize item i_6 as a true positive recommendation for user u_4 and item i_5 for users u_3 and u_4. Users u_1, u_2 and u_5 will not have any true positive recommendations since they have not interacted with any of the items. The evaluation does not consider that the items might actually be liked by the user, if recommended in a real-world situation.

Traditional evaluation estimates the users' taste by analyzing their histories of item interactions, e.g., items they have rated, purchased, or otherwise consumed. This type of evaluation models the accuracy of the recommender algorithm, to a certain point [22]. In order to further estimate the quality from the user's perspective, a different set of metrics and evaluational concepts need to be considered instead.

11.2.5 Multi-dimensional Evaluation and Benchmarking

In any real-world application, recommendation systems should simultaneously satisfy (1) functional requirements that relate to qualitative assessment of recommendations, and (2) non-functional requirements that are specified by the technological parameters and business goals of the service. These requirements have to be evaluated together: without the ability to provide sufficiently accurate recommendations, no recommendation system can be valuable. Since bad quality recommendations will have an adverse effect on customer retention and user loyalty, they ultimately will not serve the business goal of the service (see Sect. 11.2.5). Similarly, if the recommendation system does not scale well with the characteristics of a service, and is not able to provide recommendation in real time (the response

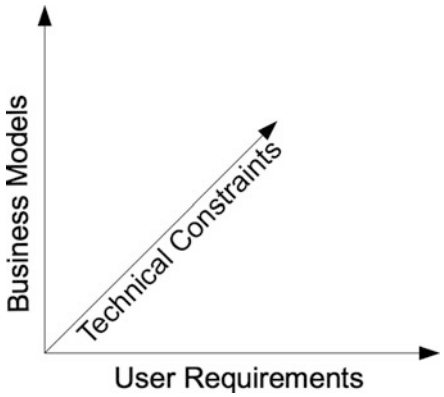

Fig. 11.2 The three primary dimensions of the multi-dimensional evaluation approach

time depends on the virtue of the service, but usually ranges within 10–1,000 ms), neither users nor service provider benefit from the recommender. Consequently, a tradeoff between these types of requirements is necessary for an impartial and comprehensive evaluation (and if needed, benchmarking) of real-world recommender solutions.

The three primary dimensions of the multi-dimensional evaluation approach [37] are shown in Fig. 11.2.

User Aspects

From the user perspective, the benefits of recommendation systems lie in their persuasiveness, i.e., their capability to influence a user's attitude, decisions, or behavior. By making it easy to access information, and by tailoring the content offered, a recommender can affect a user's attitude positively toward the application, and make their relationship with the system more trustful. According to the user's tasks, recommendations may have different goals. The goal could, for instance, be to reduce information overload, facilitate search and exploration, and identify interesting items, increasing the quality and decreasing the time of the decision-making process. Increased trust and confidence in the service could also be key factors [23].

Perception of quality is a broad concept commonly discussed in topics ranging from e-business, where Lai [27] developed methods and instruments for measuring the perceived quality of electronic services among employees of several international electronic services; to e-learning, where Sumner et al. [42] identified educators' expectations and requirements on educational systems; to information quality, where Goh et al. [19] compared perceptions of the users' engagement in a mobile game, etc. These works show that, in the context of recommendation systems, perception of quality is not only dependent on the accuracy of the recommendation but also on other, domain-dependent, factors.

Different ways of presenting recommendations can also result in different perceptions based on cultural settings. This has been shown in different contexts, e.g., Chen and Pu [11] show this within a user study ($n = 120$) on participants from a "western culture" and participants from an "oriental culture" that was performed in order to evaluate, among other issues, the perceived quality of an organizational recommendation system. The study showed that even though cultural differences do not affect the complete spectrum of perception-related concepts, the perceived quality in one out of two presentation formats did differ significantly between the cultures. Similar concepts in information interaction were studied by Barber and Badre [5], showing that some design elements in websites are perceived differently across cultures.

Similarly Cremonesi et al. [13] compared the quality of recommendation algorithms in seven different systems by means of a user study ($n = 210$). The three principal findings were that (1) non-personalized recommendation algorithms provided for high user satisfaction, although with low utility; (2) content-based algorithms performed on par with, or better than, collaborative filtering-based recommendation algorithms; and (3) traditional accuracy metrics (recall and fallout) did not approximate the perceived quality very well.

It seems clear that the user's perception of the recommender does not need to be tied to the actual measured performance of the recommendation algorithm. Instead, the context of the recommendation dictates how it will be perceived by the end user. It is for this reason that the user aspects need to be prioritized, in contexts where they are important. Recommendation accuracy, in its traditional sense, is only important in some of these contexts. Others can stipulate that the recommendation accuracy can very well be low, as long as other factors attain desired levels.

Business Aspects

The business model is the method that allows a company to generate revenue and to sustain itself. Different business models may lead to different requirements in terms of expected added value from a recommendation system. Examples of business requirements include: increased profit, increased revenues, increased user retention and user loyalty. For instance, in a pay-per-view video-on-demand business model (see Sect. 11.1), the goal of the recommendation system may be to increase sales of movies that allow the company to maximize revenues (e.g., movies with the largest cost and/or margin). However, in subscriber-based video-on-demand business model, the driving forces for a company may be the desire to get users to return to the service in the future, i.e., increase user retention and customer lifetime; a typical showcase where recommendation systems help is outlined by Dias et al. [16].

One can also differentiate between recommendation scenarios when the recommendation system has direct effect on the revenue generated or the influence is rather indirect. Typical examples of the first case include webshop recommendation and video-on-demand recommendation. There, the process of converting

a recommendation to an actual purchase usually contains multiple steps, and the recommendations should be both relevant and persuasive for successful purchase conversion. First, the user should click on the recommended product, then add it to the cart, and finally confirm the purchase at the checkout.[8] Accordingly, the success of recommendation is measured and evaluated in each step, since the user can churn at any stage of the purchase process. First, the click-through rate (CTR) of the recommendations is measured: the ratio of successful recommendation box displays, i.e., when the user clicks on a recommended content compared to the total number of recommendation box displays. Second, the conversion rate is measured: the ratio of recommended content views that result in a purchase (Fig. 11.3).

In other recommendation scenarios, for instance in video streaming and "tube" sites like YouTube, or in news websites, the goal of the recommendation system is to keep the users on-site by providing relevant, perhaps also serendipitous and usually novel content for them. The business model of such sites is usually advertising and page view driven: the more page views attained, the more advertising surface can be sold, hence increasing the revenue of the site. Page views can be increased primarily in each user visit, or by increasing the frequency of user visits. Consequently, the primary business evaluation metric is the CTR of recommendations, and additionally the average page views per visit, average user page per days/months, or return frequency of users.

Video streaming sites like Hulu may also sell advertising units within the content: depending on the placement of the advertisement, one can talk about pre-, mid-, and post-roll advertisements (i.e., temporally relative to when the actual content is shown). In such cases, the success of recommendations can be also quantified by the average video watch length compared to the total video length, or by the number of in-video ad views achieved by the recommendations. Therefore, successful recommendations should not just have an interesting and appealing title or thumbnail image, but their content should also be relevant for the user.

CTR measured on recommendation boxes is higher than the conversion rate of the entire recommendation conversion chain. As the conversion rate rather evaluates the overall success of the whole system and since many factors independent from the quality of recommendations may be influential, the CTR on the recommendation boxes should be considered as the direct success measure of the recommendations themselves [e.g., 33]. The influence of relevant and persuasive recommendations, however, reaches beyond the first clicks and result in higher conversion rate as well. Interestingly, Zheng et al. [47] showed that empirical CTR values and the relevance of recommendations are not consistent; thus, using only CTR as the ultimate metric for online evaluation of recommendation systems may be biased and restrictive. They also suggest that optimizing recommendations based on normalized Google distance [12] may be a better proxy for evaluating

[8] In some webshop implementations, clicking on a recommended content can directly add the content to the cart, thus reducing the number of steps and simplifying the purchase process.

Fig. 11.3 CTR and conversion rate. In this case the conversion rate is calculated as the product of the three metrics: CTR, add-to-cart CTR, and checkout CTR

recommendation relevance than CTR, since the latter is biased by content popularity.

Ideally, the evaluation metrics for business aspects should correlate and be consistent with traditional offline evaluation metrics as discussed above. However, both research from academia and practitioners from industry realize the gap between

offline and online evaluation settings [20]. With sales and profitability as the obvious common baseline denominator, the specific core metrics for real-world application are use-case dependent, while academic research tends to focus on well-defined recommendation related problems that can be assessed by standard offline error based or information retrieval metrics.

Technical Aspects

In addition to user-centric and business-related requirements that relate to qualitative assessment of recommendations and the business rationale, in any real-world application recommendation systems should equally meet non-functional requirements that are specified by the technological parameters. The choice of candidate recommendation systems for specific real-life applications must take into account a number of technical requirements and constraints including data and system constraints, as well as reactivity, scalability, adaptability, and robustness.

Data constraints derive from the communication architecture. For instance, aerial or satellite TV services lack a return channel for transferring user feedback, hindering the application of traditional collaborative filtering algorithms. As a second example, linear TV services typically lack good-quality metadata because of the large amount of video content produced and broadcast every day. In this scenario, content-based filtering techniques could not be applied.

System constraints derive from hardware and/or software limitations in the service provider infrastructure. For instance, in a mobile TV scenario, the processing and memory capacity in the users' hand-held devices are limited and algorithms requiring significant computation or storage on the client side cannot be applied.

Reactivity is understood in a recommendation system as the ability to provide good quality recommendations in real-time where the time threshold depends on the application area use case, typically in the range of 10–1,000 ms. In an online setting, fast response time is a must, because web users do not tolerate slow webpage load times [30]; moreover, it has also been shown that load performance correlates strongly with shopper conversion and bounce rate. Therefore, the reactivity of the recommendation system also influences the its business success.

Although non-origin content (third party services, typically analytics, advertising and social network plug-ins) accounts for an increasing portion of the total fetched objects and bytes, interestingly, their contribution to page load time is minimal [8], which explains the popularity of using third party recommendation systems provided by specific vendors. Recommendations are typically displayed asynchronously to prevent slowing down loading of the main content; however, this can only be a partial remedy, when the entire page content is personalized, such as location-based and personalized news aggregator services [38].

Quick response time is less critical for batch recommendation tasks, like personalized newsletter generation, when a large number of recommendations should be provided for many users.

Summarizing, reactivity of online recommendation systems is measured by performance indicators including but not limited to average response time, and response rate exceeding the time threshold is measured.

Scalability of recommendation systems is generally understood as the ability to provide good quality recommendations independently of the size of the dataset and its primary dimensions (number of user and items), its growth, and the dynamic of the growth. Scalability requirements can be further broken down into model initialization and maintenance-related (training and updates) and online-operation-related parts. By the former is meant the ability to process extremely large, potentially heterogeneous datasets at the system initialization and recurrent system update phases (including model building if necessary) using computational resources linearly scalable with the data size. The latter is meant as the ability to serve large amounts of parallel recommendation requests in real time (see also reactivity) without significant degradation in recommendation quality. In other words, online scalability extends the concept of reactivity for many simultaneous user accesses.

The online scalability of initialized recommendation systems can be validated through stress-tests where recommendation requests are sent in scalable multi-thread configuration, and scalability performance indicators—e.g., recommendation throughput, response success rate, fallback response—are measured.

These requirements are particularly strict in linear TV applications, where millions of TV viewers are used to a very responsive interface. In this scenario, there is the need to use recommender algorithms able to run (and make efficient usage of all resources) on a multi-processor and multi-node distributed environment. As shown by Takács et al. [43], memory-based collaborative filtering algorithms may fall short in this scenario.

Adaptability of recommendation systems is the ability to react to changes in user preferences, content availability and contextual parameters. Adaptability is crucial to overcome the cold start problem. Adaptive recommender algorithms are able to capture new users' preferences after the first few interactions, and thus the quality of recommendations is improved by each user click. Analogously, adaptive item modeling is of particular importance for new items, since integrating user feedback on new items may improve recommendation quality significantly, when metadata is not sufficient for appropriate item modeling [34]. Therefore adaptability can be measured as the recommendation quality for new users and on new items. Adaptability to changes in user preferences and content availability can also be measured by systematic synthetic tests populating users and/or items incrementally to the recommendation systems. Similarly, adaptability to changes in contextual parameters will be tested by varying certain contextual parameters of recommendation requests.

Robustness *requirements* are necessary to create high-quality recommendation services, able to work in case of data corruption or component failure in a distributed environment. Such a situation may equally arise during system initialization and operational phases. These requirements are typically translated into the need for fault-tolerant recommenders able to run on high-availability clustered systems,

using, for instance, fallback components in case of system failures and including algorithmic solutions that are stable against missing or corrupted data [1].

11.2.6 When to Benchmark and When to Evaluate

In the context of this chapter, the difference between benchmarking a recommendation system and evaluating it is based on the expected outcome of the process. Evaluation is traditionally used in order to estimate the quality of a single system, i.e., using the same recommendation context, datasets, and implementation frameworks. An example of this is when tuning an algorithm to either higher accuracy, lower running time, or any other sought-after value. Benchmarking on the other hand is applied in order to compare systems not necessarily deployed in the same environment, e.g., the previously mentioned Netflix Prize. Benchmarking allows a system-to-system comparison between not only different algorithmic implementations, but across various datasets, frameworks and recommendation contexts—provided a benchmarking protocol is defined, e.g., when benchmarking a single evaluational aspect, a protocol for the evaluation metric is specified and used across different systems.

In terms of recommendation accuracy, a benchmarking protocol might specify what measures, metrics, data splits (training/validation sets) and other relevant factors to use in order to allow for a fair comparison (e.g., see Fig. 11.1). Similarly, for multi-dimensional benchmarking, it is imperative to specify such a benchmarking protocol in order to ensure a fair and accurate comparison. In this context, the protocol should include the dimensions to measure, how these should be measured (e.g., specifying metrics, how datasets should be prepared, etc.), and how the final benchmark score should be calculated in order to allow for a simple means of comparison. In benchmarking events such as the Netflix Prize or the KDD Cup, the benchmarking protocol was given by the organizers; when running a stand-alone benchmark, the protocol needs to be defined such that it meets the purpose of the comparison. An example of this is presented below.

11.3 Benchmarking Example

As discussed above, the process of benchmarking a recommendation system should be dependent on the use case it is deployed in. This applies to any measurable attribute that is to be benchmarked, whether speed, or recommendation accuracy. Failure in doing so could potentially prove detrimental to the overall quality of the recommendation system.

This section illustrates the application of the proposed benchmarking model. Three different recommendation approaches are evaluated and benchmarked, each having a different characteristic, and each showing the value of a multi-dimensional evaluation approach.

11.3.1 Evaluation Setting

In order to comprehensively evaluate a recommendation system f, multiple objectives need to be taken into consideration. In the scope of this chapter, these objectives come from the three dimensions: user aspects, business aspects, and technical constraints. Each of these is represented by some evaluation metric $E_i(f)$.

For the sake of convenience, we assume that all evaluation metrics are formulated as utility functions, which we want to maximize. We define the multi-objective evaluation function E by

$$\mathbf{E}(f) = \begin{bmatrix} E_1(f) \\ E_2(f) \\ \vdots \\ E_p(f) \end{bmatrix}, \tag{11.1}$$

where $\mathbf{E}(f)$ is a vector of evaluation metrics $E_i(f)$ and p is the number of evaluation metrics. This setting corresponds to the three-dimensional benchmarking model presented in this chapter.

The question to be answered is as follows: Suppose that we have a set of two recommendation systems, f and f'. Which of these systems is more suitable to deploy in our context, as defined by the multi-objective evaluation function $\mathbf{E}$?

The field of multi-objective optimization suggests several approaches to this question [e.g., 17,45,49]. In order to keep the example evaluation below simple, we present one common approach: weighting. This is done by combining the evaluation metrics E_i into one single, weighted, global evaluation criterion:

$$U(f) = \mathbf{w}^T \mathbf{E}(f) = \sum_i w_i \, E_i(f) \,, \tag{11.2}$$

where $\mathbf{w}$ is a column vector and $w_i \geq 0$ are weights specifying the importance of the evaluation metric E_i. Using the utility function U, a recommendation system f is seen to perform better than f', if $U(f) > U(f')$. It is however crucial that the choice of evaluation metrics E_i and weights w_i are problem-dependent design decisions: each recommendation system needs to have these specified based on its context, the users' context, and the business context, i.e., a benchmarking protocol.

11.3.2 Benchmarking Experiment

In this benchmarking experiment, we demonstrate multi-objective evaluation of three recommendation algorithms, each tuned to a specific recommendation quality. The algorithms are:

- k-nearest neighbors (kNN) is a traditional recommendation algorithm widely-used for recommendation in a wide variety of settings. kNN recommends items that are preferred by users similar to oneself, i.e., one's neighbors. This can however cause low diversity due to effects of popularity, e.g., highly rated popular movies are often recommended to very many users [10].
- k-furthest neighbors (kFN) is an algorithm that turns the kNN algorithm inside-out and recommends items that are disliked by users dissimilar to oneself. The algorithm is specifically tuned to deliver more diverse recommendations, e.g., those that traditional recommendation algorithms fail to recommend, while still keeping the recommendations personalized [36].
- Random (Rnd) is a random recommender. This recommender is non-personalized, simply recommending a random selection of the items available. The benefit of this algorithm is its constant speed, i.e., independent of the numbers of users or items. The random recommender has an obvious inherent component of diversity and novelty, although with random accuracy, which is presumably low.

For each of the three axis, the following data is available: (1) business axis: the users' intention to return to the site; (2) user axis: the usefulness of the recommendations; and (3) technology axis: the computation time required to calculate recommendations. The data is based on a user study ($n = 132$). The study was set up as a simple movie recommender (described in detail by Said et al. [36]) where users would rate a number of movies and receive recommendations based on the input. The above three recommendation algorithms were employed.

For the business and user axes, upon receiving a set of 10 recommended movies users were asked whether they would consider using the system again (intention of return), and whether the recommendations were useful (usefulness of recommendation). Answers to the questionnaire amount to ratings normalized to a scale from 0 (not appropriate) to 1 (highly appropriate). Based on these data, we selected the following evaluation metrics:

- $E_b(f)$ measures the average intention of return of f;
- $E_u(f)$ measures the average usefulness of f; and
- $E_t(f)$ measures the utility of the average computation time t_f required by f according to

$$E_t(f) = \frac{a}{1 + \exp\left(\frac{t_f}{T} - 1\right)}, \tag{11.3}$$

where $T = 30$ is the maximum time considered as acceptable, and a is a factor scaling $E_t(f)$ to 1 if $t_f = 0$. The evaluation metric E_t is one at $t_f = 0$ and approaches zero with increasing computation time.

According to (11.2), we combine the evaluation metrics to a utility function of the form

Table 11.4 Utility values $U(f)$ for different weights $\mathbf{w}$. The maximum accepted time is $T = 30\,s$

	kNN	kFN	Rnd
$E_u(f)$	0.53	0.52	0.44
$E_b(f)$	0.56	0.51	0.36
$E_t(f)$	0.80	0.80	0.99
$U(f)$ with $\mathbf{w} = (0.\bar{3}, 0.\bar{3}, 0.\bar{3})$	0.63	0.61	0.60
$U(f)$ with $\mathbf{w} = (0.6, 0.3, 0.1)$	0.57	0.55	0.47
$U(f)$ with $\mathbf{w} = (0.3, 0.6, 0.1)$	0.58	0.54	0.43
$U(f)$ with $\mathbf{w} = (0.1, 0.1, 0.8)$	0.75	0.74	0.87

$$U(f) = w_b\ E_b(f) + w_u\ E_u(f) + w_t\ E_t(f)\ . \tag{11.4}$$

The choice of $\mathbf{w}$ is shown in Table 11.4.

11.3.3 Results

The utility values for different weights $\mathbf{w}$ (shown in Table 11.4) show that kNN attains better values than kFN for two out of three evaluation metrics (E_b and E_u) and ties in one (E_t). As a consequence, the utility value U of kNN is always equal to or better than the one of kFN regardless of how the weights are chosen. In multi-objective optimization, we say that kNN is Pareto-superior to kFN [26]. As expected, when business- and user-requirements are preferred, kNN and kFN outperform the random recommender. In a use case where computation time is the most critical constraint, the random recommender outperforms the others solely based on the speed of the recommendation. In a scenario where all three axes are equally important, kNN performs best.

The results illustrate that an appropriate choice of evaluation metrics, as well as weight parameters, are critical issues for the proper design of a utility function and benchmarking protocol. This design process is highly domain- and problem-dependent. Once a proper utility function has been set up, the performance of different recommender algorithms can be objectively compared. Since current state-of-the-art recommendation methods are often optimized with respect to a single recommendation accuracy metric, introducing multi-objective evaluation functions from the different contexts of a deployed recommendation system sets the stage for constructing recommendation algorithms that optimize several individual evaluation metrics without simultaneously worsening another.

11.4 Discussion

The concepts related to evaluation, and specifically to multi-dimensional evaluation presented in this chapter provide a motivation as to why (and how) recommendation systems can be evaluated and compared to each other across different domains, datasets, and contexts—as long as the evaluation and benchmarking protocols are specified. The creation of these protocols, and specifically the combination (weights) of the dimensions of the evaluation are however not entirely trivial and need to be chosen with the recommendation requirements in mind. When these are provided, the overall quality of the recommendation system can be estimated and compared toward other recommendation algorithms, no matter the datasets, contexts, and other system-specific deployment aspects.

Benchmarking protocols need to accurately reflect the expectations and constraints of the benchmark, such as the domains in which the recommendation systems are deployed (e.g., products, code) and other aspects related to the environment in which the recommendation systems live (e.g., framework, memory, CPU). An accurate benchmarking protocol needs to be based on a thorough analysis or empirical studies of the needs and priorities of the context in which a recommendation system is to be deployed.

It should be noted that a simple one-dimensional evaluation approach will in most cases be sufficient to tune a recommendation algorithm and estimate its quality. The added cost (in terms of development) of a benchmarking protocol that can estimate the in situ quality of an algorithm could potentially be higher than an in-place evaluation of said algorithm. However, when comparing multiple algorithms across a variety of systems, the accumulated cost will likely be lower when using benchmarking protocols than performing in situ evaluation of each candidate algorithm.

11.5 Conclusion

In this chapter, we have introduced concepts related to evaluation and benchmarking of recommendation systems, e.g., reactivity, scalability, adaptability, business values, etc. The combination of these concepts allows for a cross-system comparison of recommendation systems in order to find the most suitable recommendation algorithm for a specific recommendation context. Combined, the concepts create a benchmarking model—a protocol—that can be tuned to the specific use case of the recommender in order to accurately reflect the system's quality.

Additionally, the chapter surveyed benchmarking events such as the Netflix Prize, which set the standard for recommendation system evaluation during the last decade. Moreover, an overview of common datasets and frameworks for recommendation and evaluation was provided in order to show factors that can affect evaluation, e.g., data sparsity, size, and implementation differences across programming

languages. Following this, the chapter introduced aspects of recommendation system evaluation related to the technical constraints and business values in a deployed system, e.g., the importance of rapidly changing recommendations in a system with ephemeral items (live TV), or the importance of delivering the right recommendation not only from the user's perspective but also from the provider's (Internet marketplaces). These factors, even though seldom used for evaluation, define whether a recommendation system will be able to perform adequately in its deployed context or not.

Finally, the chapter introduced a multi-dimensional benchmarking model that allows for a comparison of recommendation systems across domain-, dataset-, and recommendation-contexts. The model takes into consideration not only traditional evaluation methods (accuracy, rating prediction error), but also any number of factors from other domains (business and technical) in order to create a simple comparable value encompassing all relevant evaluational aspects and domains.

The model allows a comparison of the qualities of recommendation systems deployed in different domains, using different datasets and having different requirements by using a tailored benchmarking protocol.

Acknowledgments The authors would like to thank Martha Larson from TU Delft, Brijnesh J. Jain from TU Berlin, and Alejandro Bellogín from CWI for their contributions and suggestions to this chapter.

This work was partially carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 246016.

References

1. Adomavicius, G., Zhang, J.: Stability of recommendation algorithms. ACM Trans. Inform. Syst. **30**(4), 23:1–23:31 (2012). doi:10.1145/2382438.2382442
2. Amatriain, X., Basilico, J.: Netflix recommendations: Beyond the 5 stars (Part 1)—The Netflix tech blog. URL http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html (2012) Accessed 9 October 2013
3. Avazpour, I., Pitakrat, T., Grunske, L., Grundy, J.: Dimensions and metrics for evaluating recommendation systems. In: Robillard, M., Maalej, W., Walker, R.J., Zimmermann, T. (eds.) Recommendation Systems in Software Engineering, Chap. 10. Springer, New York (2014)
4. Bajracharya, S.K., Lopes, C.V.: Analyzing and mining a code search engine usage log. Empir. Software Eng. **17**(4–5), 424–466 (2012). doi:10.1007/s10664-010-9144-6
5. Barber, W., Badre, A.: Culturability: The merging of culture and usability. In: Proceedings of the Conference on Human Factors & the Web, Basking Ridge, NJ, USA, 5 June 1998
6. Bell, R., Koren, Y., Volinsky, C.: Chasing $1,000,000: How we won the Netflix Progress Prize. ASA Stat. Comput. Graph. Newslett. **18**(2), 4–12 (2007)
7. Boxwell Jr., R.J.: Benchmarking for Competitive Advantage. McGraw-Hill, New York (1994)
8. Butkiewicz, M., Madhyastha, H.V., Sekar, V.: Understanding website complexity: Measurements, metrics, and implications. In: Proceedings of the ACM SIGCOMM Conference on Internet Measurement, pp. 313–328, Berlin, Germany, 2 November 2011. doi:10.1145/2068816.2068846

9. Carenini, G.: User-specific decision-theoretic accuracy metrics for collaborative filtering. In: Proceedings of the International Conference on Intelligent User Interfaces, San Diego, CA, USA, 10–13 January 2005
10. Celma, Ò., Lamere, P.: If you like the Beatles you might like ...: A tutorial on music recommendation. In: Proceedings of the ACM International Conference on Multimedia, pp. 1157–1158. ACM, New York (2008). doi:10.1145/1459359.1459615
11. Chen, L., Pu, P.: A cross-cultural user evaluation of product recommender interfaces. In: Proceedings of the ACM Conference on Recommender Systems, pp. 75–82, Lousanne, Switzerland, 23–25 October 2008. doi:10.1145/1454008.1454022
12. Cilibrasi, R.L., Vitányi, P.M.B.: The Google similarity distance. IEEE Trans. Knowl. Data Eng. **19**(3), 370–383 (2007). doi:10.1109/TKDE.2007.48
13. Cremonesi, P., Garzotto, F., Negro, S., Papadopoulos, A.V., Turrin, R.: Looking for "good" recommendations: A comparative evaluation of recommender systems. In: Proceedings of the IFIP TC13 International Conference on Human–Computer Interactaction, Part III, pp. 152–168, Lisbon, Portugal, 5–9 September 2011. doi:10.1007/978-3-642-23765-2_11
14. Cremonesi, P., Garzotto, F., Turrin, R.: Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. ACM Trans. Interact. Intell. Syst. **2**(2), 11:1–11:41 (2012). doi:10.1145/2209310.2209314
15. Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 107–144. Springer, Boston (2011). doi:10.1007/978-0-387-85820-3_4
16. Dias, M.B., Locher, D., Li, M., El-Deredy, W., Lisboa, P.J.G.: The value of personalised recommender systems to e-business: A case study. In: Proceedings of the ACM Conference on Recommender Systems, pp. 291–294, Lousanne, Switzerland, 23–25 October 2008. doi:10.1145/1454008.1454054
17. Ehrgott, M., Gandibleux, X. (eds.): Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys. Kluwer, Boston (2002). doi:10.1007/b101915
18. Fraser, G., Arcuri, A.: Sound empirical evidence in software testing. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 178–188, Zurich, Switzerland, 2–9 June 2012. doi:10.1109/ICSE.2012.6227195
19. Goh, D., Razikin, K., Lee, C.S., Chu, A.: Investigating user perceptions of engagement and information quality in mobile human computation games. In: Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 391–392, Washington, DC, USA, 10–14 June 2012. doi:10.1145/2232817.2232906
20. Gomez-Uribe, C.: Challenges and limitations in the offline and online evaluation of recommender systems: A Netflix case study. In: Proceedings of the Workshop on Recommendation Utility Evaluation: Beyond RMSE, *CEUR Workshop Proceedings*, vol. 910, p. 1 (2012)
21. Gunawardana, A., Shani, G.: A survey of accuracy evaluation metrics of recommendation tasks. J. Mach. Learn. Res. **10**, 2935–2962 (2009)
22. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inform. Syst. **22**(1), 5–53 (2004). doi:10.1145/963770.963772
23. Hu, R.: Design and user issues in personality-based recommender systems. In: Proceedings of the ACM Conference on Recommender Systems, pp. 357–360, Barcelona, Spain, 26–30 Septembert 2010. doi:10.1145/1864708.1864790
24. Jambor, T., Wang, J.: Optimizing multiple objectives in collaborative filtering. In: Proceedings of the ACM Conference on Recommender Systems, pp. 55–62, Barcelona, Spain, 26–30 Septembert 2010. doi:10.1145/1864708.1864723
25. Koenigstein, N., Dror, G., Koren, Y.: Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In: Proceedings of the ACM Conference on Recommender Systems, pp. 165–172, Chicago, IL, USA, 23–27 October 2011. doi:10.1145/2043932.2043964
26. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. ACM **22**(4), 469–476 (1975). doi:10.1145/321906.321910

27. Lai, J.Y.: Assessment of employees' perceptions of service quality and satisfaction with e-business. In: Proceedings of the ACM SIGMIS CPR Conference on Computer Personnel Research, pp. 236–243, Claremont, CA, USA, 13–15 April 2006. doi:10.1145/1125170.1125228
28. Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: Proceedings of the International Conference on Intelligent User Interfaces, pp. 31–40, Hong Kong, China, 7–10 February 2010. doi:10.1145/1719970.1719976
29. McNee, S., Lam, S.K., Guetzlaff, C., Konstan, J.A., Riedl, J.: Confidence displays and training in recommender systems. In: Proceedings of the IFIP TC13 International Conference on Human–Computer Interactaction, pp. 176–183, Zurich, Switzerland, 1–5 September 2003
30. Nah, F.F.H.: A study on tolerable waiting time: How long are Web users willing to wait? Behav. Inform. Technol. **23**(3), 153–163 (2004). doi:10.1080/01449290410001669914
31. Netflix Prize: The Netflix Prize rules (2006). URL http://www.netflixprize.com/rules. Accessed 9 October 2013
32. Perry, R., Lancaster, R.: Enterprise content management: Expected evolution or vendor positioning? Tech. rep., The Yankee Group (2002)
33. Peška, L., Vojtáš, P.: Evaluating the importance of various implicit factors in E-commerce. In: Proceedings of the Workshop on Recommendation Utility Evaluation: Beyond RMSE, *CEUR Workshop Proceedings*, vol. 910, pp. 51–55, Dublin, Ireland, 9 September 2012
34. Pilászy, I., Tikk, D.: Recommending new movies: Even a few ratings are more valuable than metadata. In: Proceedings of the ACM Conference on Recommender Systems, pp. 93–100, New York, NY, USA, 23–25 October 2009. doi:10.1145/1639714.1639731
35. Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: Proceedings of the ACM Conference on Recommender Systems, pp. 157–164, Chicago, IL, USA, 23–27 October 2011. doi:10.1145/2043932.2043962
36. Said, A., Fields, B., Jain, B.J., Albayrak, S.: User-centric evaluation of a K-furthest neighbor collaborative filtering recommender algorithm. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work, pp. 1399–1408, San Antonio, TX, USA, 23–27 February 2013. doi:10.1145/2441776.2441933
37. Said, A., Tikk, D., Shi, Y., Larson, M., Stumpf, K., Cremonesi, P.: Recommender systems evaluation: A 3D benchmark. In: Proceedings of the Workshop on Recommendation Utility Evaluation: Beyond RMSE, *CEUR Workshop Proceedings*, vol. 910, pp. 21–23, Dublin, Ireland, 9 September 2012
38. Sarwat, M., Bao, J., Eldawy, A., Levandoski, J.J., Magdy, A., Mokbel, M.F.: Sindbad: A location-based social networking system. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 649–652, Scottsdale, AZ, USA, 20–24 May 2012. doi:10.1145/2213836.2213923
39. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval, pp. 253–260, Tampere, Finland, 11–15 August 2002. doi:10.1145/564376.564421
40. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: CROC: A new evaluation criterion for recommender systems. Electron. Commerce Res. **5**(1), 51–74 (2005). doi:10.1023/B:ELEC.0000045973.51289.8c
41. Schütze, H., Silverstein, C.: Projections for efficient document clustering. In: Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval, pp. 74–81, Philadelphia, PA, USA, 27–31 July 1997. doi:10.1145/258525.258539
42. Sumner, T., Khoo, M., Recker, M., Marlino, M.: Understanding educator perceptions of "quality" in digital libraries. In: Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 269–279, Houston, Texas, USA, 27–31 May 2003. doi:10.1109/JCDL.2003.1204876
43. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res. **10**, 623–656 (2009)

44. Terveen, L., Hill, W.: Beyond recommender systems: Helping people help each other. In: Carroll, J.M. (ed.) Human–Computer Interaction in the New Millennium. Addison-Wesley, New York (2001)
45. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. Evol. Comput. **8**(2), 125–147 (2000). doi:10.1162/106365600568158
46. Walker, R.J.: Recent advances in recommendation systems for software engineering. In: Proceedings of the International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, *Lecture Notes in Computer Science*, vol. 7906, pp. 372–381. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38577-3_38
47. Zheng, H., Wang, D., Zhang, Q., Li, H., Yang, T.: Do clicks measure recommendation relevancy?: An empirical user study. In: Proceedings of the ACM Conference on Recommender Systems, pp. 249–252, Barcelona, Spain, 26–30 September 2010. doi:10.1145/1864708.1864759
48. Ziegler, C.N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the International Conference on the World Wide Web, pp. 22–32, Chiba, Japan, 10–14 May 2005. doi:10.1145/1060745.1060754
49. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Comput. **8**(2), 173–195 (2000). doi:10.1162/106365600568202