Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 13, no. 8, December 2017, pp. 1246-1256
DOI: 10.23940/ijpe.17.08.p7.12461256

Performance Evaluation of Recommender Systems

Mingang Chen[a,b], Pan Liu[c,*]

[a]*Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai 201112, China*
[b]*Shanghai Development Center of Computer Software Technology, Shanghai 201112, China*
[c]*Shanghai Business School, Shanghai 201112, China*

Abstract

Recommender systems play an important role in e-commerce. This paper discusses three classical methods - offline analytics, user study, and online experiment - to evaluate the performance of recommender systems and also analyzes their application scenarios. Some performance evaluation metrics of recommender systems are reviewed and summarized from four perspectives (machine learning, information retrieval, human-computer interaction and software engineering) combined with the above three evaluation methods. These evaluation methods and evaluation metrics summarized in the paper provide the designers with guidance for the comprehensive evaluation and selection of recommended algorithms.

Keywords: recommender system; performance evaluation metrics; machine learning; information retrieval

(Submitted on October 29, 2017; Revised on November 17, 2017; Accepted on December 1, 2017)

1. Introduction

With the rapid development of the Internet, human beings are moving towards an era of information overload from the age of information scarcity. Recommender systems are considered as one of the most effective techniques for solving the problem of information overload. Recommender systems are a class of technology that determines the users' current needs or interests in terms of their historical behaviors, social relationships, interests, and contextual information [1]. Technically, recommender systems span multiple research fields like machine learning, information retrieval, human-computer interaction, data mining, and E-commerce etc. From the application point of view, nowadays these systems have penetrated into all aspects of people's lives [13].

E-commerce is the most successful application area of recommender systems. Amazon provides a personalized recommendation for each ordinary user, such as the recommendation of similar goods, combination recommendation of goods, and reviews of goods and so on. It was reported that 35% of Amazon's sales revenue comes from the recommendation service section. YouTube recommends videos for users according to their historical interests. Netflix recommends movies that are similar to the movies users used to love. Personalized recommender systems are also widely used in news recommendation, friends' recommendation from social networking such as Weibo, and internet advertising.

Recommender systems usually consist of four core modules: collecting of users' behaviors (searching for items, purchasing of items or reviewing of items), predicting users' preferences by recommender system model/algorithm, sorting and recommendation of items, and evaluating of the recommender system, as is shown in Figure 1. Recommendation algorithm can be roughly divided into item-based collaborative filtering recommendation [5,17], user-based collaborative filtering recommendation [6,10], content-based recommendation [18], knowledge-based recommendation and hybrid recommendation [22].

* Corresponding author.
E-mail address: panl008@163.com

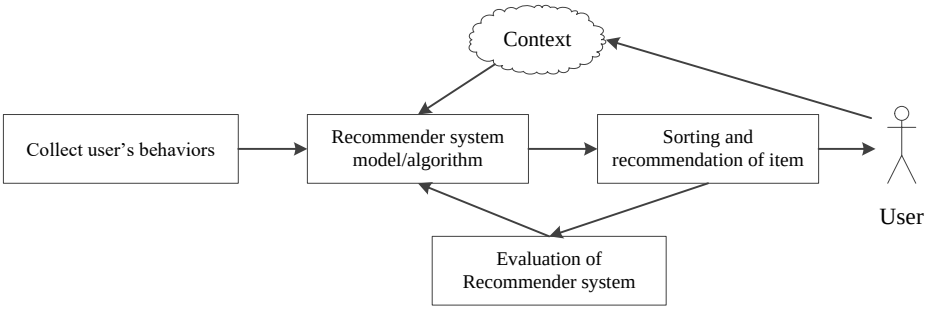

Figure 1. The module diagram for recommender system's design and evaluation

The core of recommender systems is the recommendation algorithm, but what is a good recommendation algorithm? This is the problem that needs to be studied in the performance evaluation of recommendation algorithm. In fact, the performance evaluation of recommendation algorithm is the basis of algorithm selection. Any practical recommendation algorithm needs to be evaluated by various datasets to obtain the optimal parameters before deploying to the online system. However, the performance evaluation of the recommender systems is still one of the challenges in the research field of recommender systems [11,26]. The main reasons for this challenge are as follows:

- The performance of recommendation algorithms is different on datasets with different data scales. For example, some algorithms may perform well for small datasets, but as the scale of dataset grows, the accuracy and speed of the algorithms may decrease significantly.
- There are many evaluation indicators for the recommender systems, and some of the indicators are inherently contradictory, such as the accuracy indicator and the diversity indicator. Improving the diversity of the recommended items often reduces the accuracy of recommendation.
- Different evaluation indicators of recommender systems need to be measured by different testing and evaluation methods. For example, Prediction Accuracy can be computed by offline analytics, but serendipity of recommended items needs to be obtained through user studies, and some indicators are obtained through online experiments.

Therefore, on one hand, the performance evaluation of recommender systems can help researchers analyze and evaluate the advantages and disadvantages of various recommendation algorithms objectively, and on the other hand, it can be applied to performance tuning of the systems.

This paper first introduces three performance evaluation methods of recommender systems and analyzes the advantages and disadvantages of these methods, as well as the factors that need to be considered in designing these methods. Secondly, the paper analyzes evaluation metrics of recommender systems from four perspectives: machine learning, information retrieval, human-computer interaction and software engineering. The application scenarios of these metrics are also studied. Finally, the paper summarizes metrics of recommender systems with three evaluation methods.

2. Evaluation methods of recommender systems

In general, a new recommender system needs to complete the performance evaluation of three stages before it is finally launched: offline analytics, user study, and online experiment. Offline analytics does not require user interaction, and only needs to use datasets to calculate the corresponding evaluation metrics, such as the prediction accuracy and coverage. Offline analytics is the easiest way to implement and cost the least among the three types of methods. User study requires testers to use the recommender system, perform a series of tasks, and then answer a set of questions about their experiences on the system, and finally the results of evaluation will be given through statistical analysis. Online experiment executes a large-scale experiment on a deployed recommender system. It evaluates the recommender system by the real tasks executed by the real users. The evaluation results of the online experiment are the closest to the real situations when the recommender system runs online.

2.1. Offline analytics

When making offline analytics, the dataset of the users' behaviors should be collected in advance in the first place, such as the dataset of the choices or ratings made by users on the items. And these datasets can be used to simulate the interactions between users and the recommender systems. In practice, there are two strategies which can be considered to prepare for the datasets. First is dataset from the logs of users' behaviors sampled randomly, and second is the entire log dataset before a certain time stamp [16]. It is worth noting that the collected dataset of users' behaviors should be similar to the true

behaviors of the users' interaction on the recommender systems. That is to say, the dataset used for offline analytics should be as close as the dataset to be generated from the systems after they are deployed online. In terms of statistics, the users' behaviors used for offline analytics and evaluation should be as far as possible unbiased. Only based on the above postulated conditions, the offline analytics can make reliable evaluations on the performance of the recommender systems.

The basic method of offline analytics on recommender systems is also the common practice in machine learning, which divides the dataset into the training dataset and testing dataset, and then constructs recommendation models on the training dataset and tests its performance on the testing dataset [8]. The testing method usually will be k-fold cross-validation. In fact, the core idea of offline analytics comes from the evaluations on the algorithms of machine learning and information retrieval.

The advantage of offline analytics is that it doesn't need the interaction from real users, so it can be implemented at a low cost and can test and evaluate the performance of different kinds of recommendation algorithms quickly. But the disadvantages are that such experiments can usually be used in evaluating the prediction accuracy of the algorithms or Top-N precision of recommendation, and can do little in the evaluation of serendipity or novelty and so on [20].

The main targets of offline analytics are to compare the performance of the recommendation algorithms in some metrics and to filter inappropriate algorithms and to remain some candidate algorithms. Therefore, the more costly user study or online experiment can be carried out for further evaluation and optimization.

2.2. User study

Some testers should be recruited to do user study, and be required to do some tasks using the recommender systems. When testers execute the tasks, observe and record their behaviors, and collect the situations of their tasks, such as which tasks are completed, and how much time is consumed on the tasks and the accuracy of the tasks' results. We can also require testers to answer some qualitative questions by user study, such as whether they like the UI for the users, or how they feel about the complexity of the tasks, and these results cannot be obtained in offline analytics [24].

User study is an important method for evaluating recommender systems. This method will test the interaction between users and the recommender systems, and can obtain the influence of the recommender systems on the users. User study can also be used in collecting qualitative data, and these data are of great importance in explaining the quantitative results. However, user study also has some weaknesses. First, the cost of user study is very high. In one hand, a large number of testers must be recruited and on the other hand, testers must finish a large number of interacting tasks. Therefore, in common practice, the number of testers and the size of testing tasks should be controlled. Meanwhile, the quality of the data to be collected should be guaranteed in statistical significance.

Besides, the distributions of testers should be considered, such as the distributions in hobbies and interests, sex ration, ages, activity levels, etc. should all be similar to those of the users in a real system. It's also important that, when collecting data from the users' tasks, the purposes of the testing should not be told to the testers before testing, in order to avoid the subjective tendency in users' behaviors and answers, such as accepting more recommended information unintentionally.

2.3. Online experiment

Online experiment is to execute a large-scale testing on a recommender system which is already deployed. Online experiment can be used to evaluate or compare different recommender systems by the real tasks carried out by real users. Online experiment can achieve the most real testing results among the three evaluation methods. The advantages of online experiment are that, the entire performance of the recommender systems can be evaluated, such as long-term business profit and users' retention, rather than some single metrics. Therefore, online experiment can be used to understand the impact of the evaluation metrics (such as the accuracy in prediction, diversity in recommendation) on the overall performance of the system. The balance of these metrics will be considered when choosing the parameters of the recommendation algorithms.

In many real recommendation applications, the designers of the system wish to influence the behaviors of users using recommender systems. Therefore, when the users interact with recommender systems based on different algorithms, the designer wishes to evaluate the influence of the recommender systems on users' behaviors through online experiments. Here is an example: if the user clicks only one item, when the system based on algorithm A recommends him five items,

meanwhile the user clicks four items when the system based on algorithm B also recommends him five items. Then, we regard the recommender system based on algorithm B better than the one based on algorithm A.

The actual effect of the recommender systems depend on various factors, such as the real intention of the users (for example what is their particular demand), the situations of the users (for example what are their familiar items, how much trust they have on the recommender system), and the UI of the systems.

To carry out an online evaluation, the following problems should be taken into consideration. For example, the random sampling of the users, to ensure that the distribution of users for different recommender systems is as similar as possible, and then different recommender systems can be compared fairly. Besides, as one metric is concerned, the other influence factors should be consistent. For example, if the prediction accuracy of the algorithm is concerned, then the UI of different recommender systems based on different algorithms should be the same, and if the UI is concerned, then the algorithms that different recommender systems based on should be the same [9].

It is worth noting that, there is some risk in online experiment. For example, if a recommender system recommends too many unrelated items during online experiment, then the users' trust on the recommender system will be reduced rapidly and will not care for the items being recommended after the real system is deployed finally. This is the worst and most unacceptable situation in commercial practices.

Based on the above reasons, online experiments are often placed at the last period among the three kinds of evaluation methods. Generally speaking, offline analytics is used to evaluate and compare the algorithms of different recommender systems and achieve some most appropriate candidate recommendation algorithms. Then in the user study phase, by recording various users' tasks interacting with recommender systems, evaluating the acceptance of the testers at candidate recommending algorithms, a further choice is made on candidate recommendation algorithms and the optimization of the parameters of algorithms. And at last, the most appropriate recommender system will be picked up by online experiment. The progressive evaluation process will reduce the risk of online experiment, and accomplish satisfying recommendation results.

3. Performance evaluation metrics of recommender systems

3.1. Perspective of machine learning

In the literature of recommender systems, a variety of machine learning algorithms were used to predict user ratings, including regression, SVD, PCA, probability inference, and neural network and so on [28,32]. Therefore, the prediction accuracy from the perspective of machine learning is naturally applied to the evaluation of the accuracy of users' ratings of recommender systems.

The metric of prediction accuracy is essentially about the error of prediction. This metric is a common metric in evaluating various machine learning algorithms, such as regression or classification. This metric used in evaluating recommender systems is mainly used to measure the ability to predict users' behaviors. Prediction accuracy is the most important metric in the offline analysis of recommender systems. This metric is commonly used in early papers of recommender systems researches to discuss the accuracy of different recommendation algorithms.

When calculating prediction accuracy, you need a set of offline dataset that contains users' scores, such as users' ratings for a product or movie. The dataset is divided into training set and testing set. A users' rating prediction model is trained on the training set, then the prediction of users' rating is computed on the testing set. The error is the deviation between the predicted rating and the real rating. There are three metrics to measure the prediction accuracy: Mean Absolute Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE), and the formulas are as follows:

Mean Absolute Error:

$$MAE = \frac{1}{|Q|} \sum_{(u,i)\in Q} |r_{ui} - \hat{r}_{ui}| \tag{1}$$

Mean Square Error:

$$MSE = \frac{1}{|Q|} \sum_{(u,i)\in Q} (r_{ui} - \hat{r}_{ui})^2 \tag{2}$$

Root Mean Square Error:

$$RMSE = \sqrt{\frac{1}{|Q|} \sum_{(u,i)\in Q} (r_{ui} - \hat{r}_{ui})^2} \tag{3}$$

where Q is the test set, r_{ui} represents the user's true ratings, $\hat{r}_{ui}$ represents the prediction ratings of the recommender system. MAE is the simplest, but it does not take into account the direction of the error (positive error or negative error). MSE has a larger penalty on large errors and the squared error does not have an intuitive meaning. Therefore, RMSE is more widely used in computing the prediction accuracy of the recommender system.

It is worth noting that while using the prediction accuracy to compare the recommendation algorithms, it is necessary to use the same dataset for different algorithms. Although the prediction accuracy is mainly concerned with predicting users' rating, this metric often has guiding significance to the overall performance of recommender system [6].

3.2. The perspective of information retrieval

Recommender systems are also regarded as a special case of information retrieval systems, which retrieve the related information from the user's historical data [14,25].

Although the prediction accuracy is an important metric of evaluating a recommender system, sometimes the user does not care about the specific numerical accuracy, such as the user does not care about whether a rating of a movie is 4.9 or 4.8; the user is concerned about whether the film is wonderful or not.

So it is necessary to study metrics of decision support and ranking. The target of decision support is to help users to choose "good" items while the metrics of ranking are concerned with the order of the recommended items that users like.

3.2.1. Precision, Recall and F-measure

A recommender system usually recommends a list of items for users. They are generally arranged in horizontal or vertical. Users generally only care about the front parts of several items and few users will care about the back parts of the items, such as the items listed on page 20. This way of recommendation is called Top-n recommendation. We can use the precision, recall and F-Measure from information retrieval field to evaluate the performance of recommender system. The formulas are as follows:

$$Precision = \frac{N_{rs}}{N_s} \text{ or } Precision@n = \frac{N_{rs@n}}{n} \tag{4}$$

$$Recall = \frac{N_{rs}}{N_r} \tag{5}$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

where $N_{rs}$the number of the recommended items that user prefer, N_s is the number of the recommended items and n is the first n recommended items. N_r the number of items that user prefer.

The precision of recommendation describes the proportion of items that users prefer, and the recall describes the proportion of the user favorite items that are not missed. F-Measure is a compromise between the precision and the recall. In the application of recommendation, users generally pay more attention to the precision of the top-n, rather than the recall.

3.2.2. Mean Average Precision

The mean average precision (MAP) is the average of multiple recommendation precision. The more precision the recommended items are, the higher the MAP is [29], and the formula of MAP is as follows:

$$MAP = \frac{\sum_{q=1}^{Q} Avep(q)}{Q} \tag{7}$$

$$Avep(q) = \frac{\sum_{k=1}^{n} p(k) \times rel(k)}{\#\ relevant\ \ item} \tag{8}$$

where Q is the number of recommendation, k is the rank, $rel(k)$ represents the relativity function given rank k, $p(k)$ represents the precision given rank k.

3.2.3. ROC Curve

ROC is an acronym for "Receiver Operating Characteristic," which was originally used as an analysis tool for signal detection. In recent years, ROC analysis has been widely used in the field of information retrieval and machine learning algorithms evaluation [7]. The ROC curve is a two-dimensional coordinate graph, the X-axis is the false positive rate (FPR) and the Y-axis is the true positive rate (TPR). The ROC curve shows the correspondence between FPR and TPR, as is shown in Figure 2.

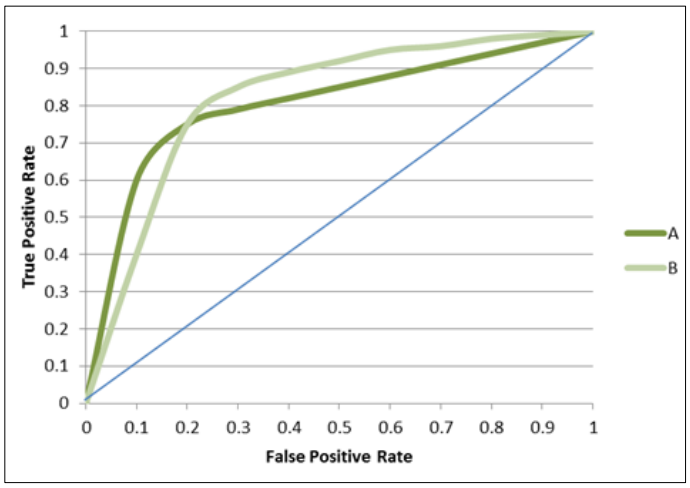

Figure 2. ROC curve

We can adjust the parameters of the recommender system through the ROC curve, such as finding the compromise between the FPR and TPR. In addition, the performance of different recommender systems can also be compared by AUC (the Area Under the ROC curve).

3.2.4. Mean Reciprocal Rank

The concept of Mean Reciprocal Rank (MRR) comes from the information retrieval system, which means the more relevant retrieval results should be ranked ahead. When metric of MRR is applied to the recommender system, it can measure whether the recommender system places the user's favorite items in the front [2]. The MRR is defined as follows:

$$MRR = \frac{\sum_{q=1}^{Q} 1/rank_i}{Q} \tag{9}$$

where Q is the number of recommendation, $rank_i$ are the user's favorite items in the list of recommendation rankings. Obviously, the larger the value of MRR is, the better the performance of the recommender system is.

3.2.5. Spearman Rank Correlation Coefficient

Spearman rank correlation coefficient (SRCC) is used to calculate the Pearson correlation coefficient between the ranking of the recommended items and the ranking of the ground truth [11,26]. The SRCC is defined as follows:

$$RCC = \frac{\sum_i (r_1(i) - \mu_1)(r_2(i) - \mu_2)}{\sqrt{\sum_i (r_1(i) - \mu_1)^2} \sqrt{\sum_i (r_2(i) - \mu_2)^2}} \tag{10}$$

where $r_1(i)$ and $r_2(i)$ the ranking of the recommended items and ground truth, and μ is the average of ranking. If the ranking of the items recommended by system is the same as the real rankings, the value of SRCC is 1.

3.2.6. Normalized Discounted Cumulative Gain

Spearman rank correlation coefficient does not actually consider the position of rankings, in other words, the penalty for all error rankings are the same. Therefore, in recent years, Normalized Discounted Cumulative Gain (nDCG) has been widely used in the evaluation of recommender systems [12]. The nDCG is defined as follows:

$$nDCG = \frac{DCG(r)}{DCG(r_{perfect})} \tag{11}$$

$$DCG(r) = \sum disc(r(i))u(i) \tag{12}$$

where $\mathrm{disc}(r(i))$ is a discount function based on the ranking, which makes the ranking of the preceding items is more important; $u(i)$ is the utility of the items in the recommendation list, such as the user's rating or click, browse or purchase. $\mathrm{DCG}(r_{\mathrm{perfect}})$ represents a perfect ranking of discounted cumulative gain.

3.2.7. Coverage

In the recommender system, the coverage of items refers to the proportion of items recommended to total items [26]. The coverage of items is defined as follows:

$$coverage = \frac{\bigcup_{u \in U} I(u)}{I} \tag{13}$$

where $I(u)$ is the number of items recommended for a user, I represents the total number of items. Coverage is an important evaluation metric of recommender systems because it can describe the ability of mining the range of recommendation items. A good recommendation system requires not only high prediction accuracy but also high coverage.

3.3. The perspective of human-computer interaction and user experiences

Recommender systems provide recommended items for users through the interactions with users. Therefore, recommender systems shouldn't be simply regarded as recommendation algorithms, but should be also evaluated from the perspectives of human-computer interaction and users' experiences [27].

3.3.1. Diversity

In Recommender systems, the definition of the diversity of items is just opposite to that of similarity. In some situations, it makes no sense to recommend similar items for the users in practice. For example, if the user has already bought a sport watch with GPS function, generally speaking, he will not be interested in other similar sport watches from other brands recommended by the recommender system. But if the recommender system can recommend him some other items, like the heart rate belt or sport music CD, it can make much better recommendation results. Hence, while designing a recommender system, not only the accuracy of prediction, but also the diversity of recommendation products should be concerned, so as to satisfy different requirements of users [30].

3.3.2. Trust

Trust means the trust level that the users have on the recommendations made by recommender systems. If the recommender system recommends some items for a user, the user is acquainted with the items, and like all the items, and then he will think that the recommender system has provided appropriate items and will trust the system. On the contrary, if the user is not interested in the items recommended by the recommender system, then he will lose trust in the system. Trust on the recommender system can be obtained from users' survey, which is enquiring on the users to know if they trust the results of the recommender system or not. At present, the main method used in recommender systems to win the trust from the users is to explain the reasons of recommendation, by telling the users why these items are recommended to him, such as another user who has the same hobbies and interests is also fond of these items, etc. The reasonable explanations can enhance the users' trust on the recommender system [23].

3.3.3. Novelty

The novelty of the recommender system means to recommend items that users are not familiar with [4]. In an application that needs to recommend novel items, the most obvious and easy-realizing method is to filter the items that the user has already bought or rated. However, in many situations, the users will not tell the recommender system that they have already known all these items. This simplest method cannot filter all the items the users already known efficiently. Another method to improve the novelty is by using the average popularity of the recommended items, and this means the more popular a recommended item is the less novelty it has and to recommend some less popular items will make the users feel novelty. At present, the novelty of the recommender system can be acquired through users' survey. A commercial recommender system should balance the prediction accuracy, the diversity and novelty in the recommendation.

3.3.4. Serendipity

In a recommender system, the serendipity means the serendipitous levels that the system can bring about [31]. For example, if a user like the films stared by a certain film star, then he will feel like it or have some novelty when the system recommends him some early films stared by the film star, but he will not feel surprised or unexpected. A random recommendation will bring serendipity to the users, but if too many unrelated items are recommended, the trust on the recommender system will be reduced. Therefore, it is becoming more and more concerning to the researchers on how to balance prediction accuracy and serendipity. To evaluate serendipity levels, first the similarity between the recommended items and the previous preferred items of the user should be defined, and second, the satisfaction of the users on the recommended items should be statistically recorded. The first step can be calculated through offline analytics and the second step should be accomplished by users' survey.

3.4. The perspective of Software engineering

In this section, we will evaluate the performance of recommender systems from the perspective of software engineering.

3.4.1. Real-time

In the era of big data, increasing attention is paid to the real-time of the recommender system, as many items have real-time traits, such as news, new arrivals, etc. If a commercial website can recommend the users their favorite new arrivals in the first time, it can win adequate web flows and gain opportunities. The real-time of the recommender system includes two aspects: one is a real-time recommendation of the newly added items in the system to the users, and the other is a real-time recommendation of new items to a user according to his or her current behaviors (like purchase or click, etc.) [19]. Frameworks based on a parallel algorithm can be considered to improve the real-time of the recommender system, such as Apache Mahout or Apache Spark MLlib.

3.4.2. Robust

On one side, robust is the stability of the recommender system when facing false information which is aimed to influence the recommendation results. The algorithms of many recommender systems based on users' behaviors (like purchase or click, etc.), therefore false changes on users' behaviors may affect the recommendation results, and such situations are also called attacks on the recommender systems [21]. For example, the manager of a hotel may wish to improve the users' rating

on his hotel, and then he can register many false users to make higher rating and positive remarks on his hotel. Meanwhile, he can also make lower rating and negative remarks on his rivals. Therefore the robust of a recommender system means its ability to resist cheating behaviors, and this is similar to the search engine's ability against cheating. With regards to such attacks, an attacking model can be built and be used for the detection of users' abnormal behaviors [3]. Besides, recommender systems can adopt a more expensive users' behaviors data such as a real purchase from users instead of just rating.

On the other side, robust is the stability of the recommender system against extreme events, such as massive users' requests in a certain period of time. This kind of robust is related to the infrastructure and architecture of the recommender systems, such as the scale and reliability of hardware, the database software, etc. The recommendation algorithms based on parallel computing, such as the recommender systems based on Hadoop MapReduce programming model, will generally have higher robustness.

3.4.3. Scalability

When a recommending system is just newly online, its database (the number of items and users) is generally small-sized, and some maybe simulating data. And this problem is also called the cold start problem for recommending systems [15]. Since the aim of designing a recommending system is to recommend satisfying items to the users based on a real database, with the increasing growth of database, the recommending algorithms may slow down and require for extra calculation or storage resources. Therefore, scalability should be considered when designing a recommending system, and resources consumption should be monitored while the system is operating. For example, recommending algorithm A may have better prediction accuracy on a small-sized database compared with algorithm B, but with the growth of database, the prediction accuracy of algorithm A becomes near to Algorithm B, meanwhile the speed of algorithm A may be much slower than algorithm B, and thus it is difficult to say whether the recommending algorithm A is better or not.

Table 1. Perspectives, Metrics and Methods of evaluating recommender systems' performance

Perspectives	**Indicators**	**Evaluation Content**	**Methods**
Machine learning	MAE MSE RMSE	Prediction Accuracy	Offline analytics
Information Retrieval	Precision@n	Precision of Recommendations	Offline analytics
	Recall		
	F-measure		
	ROC		
	MAP	Ranking precision of recommended items	Offline analytics
	MRR		
	SRCC		
	nDCG		
	Coverage	Coverage of users or items	Offline analytics
Human-computer interaction and user experience	Diversity	Diversity of recommended items	Offline analytics or user study
	Trust	User trust on the recommender system	User study
	Novelty	Novelty of recommended items	
	Serendipity	Serendipity of recommended items	
Software engineering	Real-time	Real-time performance of recommender systems	User study Or online experiment
	Robustness	Robustness of recommender systems	Online experiment
	Scalability	Scalability of recommender systems	

4. Conclusions

This paper introduces three performance evaluation methods on recommender systems, including offline analytics, user study, and online experiment. Then, the advantages and disadvantages of three performance evaluation methods are discussed. Next, the paper states the application scenarios of evaluation methods and analyzes the workflow to apply these methods. Finally, the paper analyzes and concludes the various evaluation metrics for recommender systems from the perspectives of machine learning, information searching, human-computer interaction and, and software engineering as shown in Table 1. Generally speaking, designers should choose the evaluation metrics and make comprehensive analysis according to the specific tasks and aims of recommender systems, and therefore select the most optimized recommendation algorithm for the recommendation applications.

Acknowledgements

This work was supported partly by Science and Technology Commission of Shanghai Municipality Program (No.16511101202), and the National Natural Science Foundation of China (No.61502299).

References

1. G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, 2015
2. T. Bogers and A. V. Bosch, "Recommending scientific articles using citeulike," *in Proceedings of the 2008 ACM conference on Recommender systems*, pp. 287-290, Lausanne, Switzerland, October 2008
3. R. Burke, B. Mobasher B, C. Williams and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," *in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 542-547, Philadelphia, PA, USA, August 2006
4. Ò. Celma and P. Herrera, "A new approach to evaluating novel recommendations," *in Proceedings of the 2008 ACM conference on Recommender systems*, pp.179-186, Lausanne, Switzerland, October 2008
5. M. Deshpande and G.Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143-177, 2003
6. M. D. Ekstrand, J. T. Riedl and J. A. Konstan, "Collaborative Filtering Recommender Systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81-173, 2011
7. T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861-874, Amsterdam, The Netherlands, July 2006
8. A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *The Journal of Machine Learning Research*, vol.10, pp. 2935-2962, 2009
9. C. Hayes and P. Cunningham, "An on-line evaluation framework for recommender systems," In Workshop on Personalization and Recommendation in E-Commerce, Malaga, Spain, May 2002
10. J. Herlocker, J. A. Konstan and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information retrieval*, vol. 5, no. 4, pp. 287-310, 2002
11. J. L. Herlocker, J. A. Konstan and L. G. Terveen, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5-53, 2004
12. K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422-446, 2002
13. P. B. Kantor, "Recommender systems handbook," *Springer*, Berlin, Germany, 2011
14. A. B. Kouki, "Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective," *Universidad Autónoma de Madrid*, 2012
15. X. N. Lam, T. Vu, T.D. Le, and A.D. Duong, "Addressing cold-start problem in recommendation systems," *in Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pp. 208-211, Suwon, Korea, January 2008
16. N. Lathia, S. Hailes and L. Capra, "Evaluating collaborative filtering over time," in Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation, pp. 41-42, Boston, USA, July 2009
17. G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, 2003
18. P. Lops, M. de Gemmis and G Semeraro, "Content-based recommender systems: State of the art and trends," *Recommender Systems Handbook*, Springer US, pp. 73-105, 2011
19. T. Mahmood and F. Ricci, "Learning and adaptivity in interactive recommender systems," *in Proceedings of the ninth international conference on Electronic commerce*, pp. 75-84, Minneapolis, MN, USA, August 2007
20. S. M. McNee, J. Riedl and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," *in Proceedings of CHI'06 extended abstracts on Human factors in computing systems*. ACM, pp. 1097-1101, Montréal, Québec, Canada, April 2006
21. B. Mobasher, R. Burke, R. Bhaumik and C. Williams, "Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 4, pp. 23-38, 2007

22. F. Mourão, L. Rocha and J. A. Konstan, "Exploiting non-content preference attributes through hybrid recommendation method," *in Proceedings of the 7th ACM conference on Recommender systems*, pp. 177-184, Hong Kong, China, October 2013
23. P. Pu and L. Chen, "Trust building with explanation interfaces," *in Proceedings of the 11th international conference on Intelligent user interfaces*, pp. 93-100, Sydney, Australia, January 2006
24. J. Reilly, J. Zhang, L. McGinty, P. Pu and B. Smyth, "Evaluating compound critiquing recommenders: a real-user study," *in Proceedings of the 8th ACM conference on Electronic commerce*. ACM, pp.114-123, San Diego, California, USA, June 2007
25. A. Said, A. Bellogın, A. D. Vries and B. Kille, "Information Retrieval and User-Centric Recommender System Evaluation," in Proceedings of the 21st Conference on User Modeling, Adaptation, and Personalization, Rome, Italy, June 2013
26. G. Shani and A. Gunawardana, "Evaluating recommendation systems," *Recommender systems handbook*. Springer US, pp. 257-297, 2011
27. K. Swearingen and R. Sinha, "Beyond algorithms: An HCI perspective on recommender systems," ACM SIGIR 2001 Workshop on Recommender Systems, New Orleans, USA, November 2001
28. H. Wang, N. Wang and D. Y. Yeung, "Collaborative deep learning for recommender systems," *in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1235-1244, Sydney, NSW, Australia, August 2015
29. B. Yang, T. Mei, X. S. Hua, L. Yang, S. Yang and M. Li, "Online video recommendation based on multimodal fusion and relevance feedback," *in Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 73-80, 2007
30. M. Zhang and N. Hurley, "Avoiding monotony: improving the diversity of recommendation lists," *in Proceedings of the 2008 ACM conference on Recommender systems*, pp. 123-130, Lausanne, Switzerland, October 2008
31. Y. C. Zhang, D. Ó. Séaghdha, D. Quercia and T. Jambor, "Auralist: introducing serendipity into music recommendation," *in Proceedings of the fifth ACM international conference on Web search and data mining*, pp.13-22, Seattle, Washington, USA, February 2012
32. P. Zigoris and Y. Zhang, "Bayesian adaptive user profiling with explicit & implicit feedback," *in Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 397-404, Arlington, Virginia, USA, November 2006

Mingang Chen, received Ph.D. degree from Shanghai Jiao Tong University and now he is an associate professor in Shanghai Key Laboratory of Computer Software Testing and Evaluating. His main research interests include analytics of big data, and software engineering.

Pan Liu, received the Ph.D. degree in computer application from Shanghai University in 2011. Now he is an associate professor at college of information and computer, Shanghai Business School. His main interests include software testing, model-based testing, formal method, and algorithm design.