**Collaborative Filtering Recommender Algorithms: A Survey of Evaluation Metrics**

Maliheh Izadi\* and Mahdi Jalili\*\*

*\* Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*

*\*\* School of Engineering, RMIT University, Melbourne, Australia*

*Email: mahdi.jalili@rmit.edu.au*

# Abstract

Due to the explosion of available information on the Internet, the need for effective means of accessing and processing them has become vital for everyone. Recommender systems are one of the practical tools which have helped both users to find what they may be interested in, and the producers to sell their products more efficiently. Recommender systems have found great deal of attention in both academia and industry. A recommender algorithm takes into account users-items interactions (i.e., rating (or purchase) history of users on items) and their contextual information, if available. It then provide a list of potential items for each target user, such that the users is likely to positive rate (or purchase) them. In this manuscript, we survey classical and modern recommendation algorithms. We also review evaluation metrics that can be used to assess the performance of the algorithms. We then compare the performance of the algorithms in terms of different evaluation metrics on a benchmark dataset.

**Keywords:**

Recommender systems, Collaborative filtering, Matrix factorization, Ranking, Precision, Novelty, Scalability.

# 1. Introduction

Nowadays there are many commercial and non-commercial websites on the Internet offering increasing volume of diverse products to users. Recommender systems (RSs) has become an inevitable part of the world-wide-web, due to the emergence of e-commerce, wide and fast growing range of choices for customers, diversity of preferences between users, lack of precise knowledge of their needs, and lack of keyword terms to express and use search engines to meet these demands. RS are different from other filtering systems such as a search engine in that a RS tries *to find the preference of users*, then to filter through immense volume of data, and finally to recommend the most useful objects to user. Whereas, search engines are applicable when users know what they want and indicate it using the key terms. An object represent a movie, a music track, a book, an application, a restaurant, a place for vacation, a webpage or any other item used by users. We refer to an object as an item from now on. More specifically, a personalized RS utilizes user-related information to customize the generated recommendations regarding their preference. Since the first research on RSs [[1-5](#_ENREF_1)] focused on prediction and recommendation problems using the ratings alone, it has grown to an independent field of study. Recommender algorithms predict the utility of an item to a target user, and suggest the best items regarding the user’s preferences using his/her past ratings to the available items in the system.

The Netflix[[1]](#footnote-1) competition with one million dollar for the prize[[2]](#footnote-2) was an important breakthrough in RS’ history. The goal was to improve the prediction accuracy (evaluated by Root Mean Square Error) of Netflix movie recommender, *Cinematch*, by 10 percent [[6](#_ENREF_6)]. Netflix Company provided the contestants with a dataset of 480046 users, 17770 movies and 95947878 ratings. The competition was closed in July 2009, when “BellKor’s Pragmatic Chaos” team achieved the goal through considering the time of ratings in a hybrid algorithm [[7](#_ENREF_7)]. Later, due to privacy issues of its users [[8](#_ENREF_8), [9](#_ENREF_9)], Netflix removed the dataset from the Internet; however, the subsets of the original dataset are still used as benchmarks in RS studies. RS has attracted researchers from diverse disciplines other than computer science, including social science, economics, psychology, mathematics and physics.

One can look at the popularity of RS from different points of view such as business owners’ or customers’ views. An online business can amplify the income through providing its customers with what they need, along increasing the customers’ satisfaction, and gaining their trust and loyalty. Moreover, it will increase the cross-sell by recommending additional but interesting items to customers [[10](#_ENREF_10), [11](#_ENREF_11)]. On the other hand, users usually enjoy a good RS that surprises them by novel yet pleasant suggestions which they did not expect or feel the need to search for. Using an appropriate RS, they would get to know similar items to what they love or save their time finding what they want in the enormous volume of data available on the Internet. Due to increasing popularity of RS, different techniques and algorithms has been proposed to improve their performance [[2](#_ENREF_2), [3](#_ENREF_3), [11-15](#_ENREF_11)], some of which implemented in real systems. For instance, one can name Grundy system [[16](#_ENREF_16)], Tapestry system, which recommended documents from newsgroups [[17](#_ENREF_17)], Bellcore web-based Video Recommender [[3](#_ENREF_3)], Ringo, a music recommender using email [[2](#_ENREF_2)], Grouplens, for recommending Usenet news [[12](#_ENREF_12)] and Movielens movie recommender [[18](#_ENREF_18), [19](#_ENREF_19)].

Despite all the progress on RS, they encounter several challenges and need further improvement. There is a need for algorithms with better quality of recommendation, e.g. algorithms exploiting more details such as contextual information (e.g. time and location), algorithms with more flexible results or recommenders utilizing multi-criteria ratings. Moreover, there is a need for scalable algorithms which are applicable in real-world situations with large-scale datasets. As more RS algorithms are developed, it is necessary to put efforts on developing proper evaluation methods. Indeed, evaluating performance of RSs is one of the challenging tasks within the community. There is no a single metric that can efficiently measure the performance and often one has to use a number of them at the same time. For example, accuracy and diversity of RSs are not often in the same direction and one has to show both of them at the same time. Although there are a number of review articles for RSs, to the best of our knowledge, there is no comprehensive work comparing different algorithms on available evaluation metrics. In this work we aim at applying a number of well-known recommenders on a benchmark dataset and compare different evaluation metrics.

# 2. Recommender Systems

The tasks of a RS is either *prediction* or *recommendation*. Prediction means deciding whether or not a user would like a particular item by predicting *how much* she/he would rate a new item. On the other hand, recommendation refers to the task of recommending a *list* of items that the user would probably like [[20](#_ENREF_20)]. Real-world example of items for recommendations include movies [[18](#_ENREF_18), [19](#_ENREF_19), [21-24](#_ENREF_21)], music [[25-32](#_ENREF_25)], documents and books [[33-40](#_ENREF_33)], news [[12](#_ENREF_12), [41-45](#_ENREF_41)], jokes[[46](#_ENREF_46)], e-learning material [[47-52](#_ENREF_47)] and e-commerce applications [[13](#_ENREF_13), [53-60](#_ENREF_53)]. RS utilize various explicit or implicit information sources to generate their recommendations, including ratings, social information, demographic and contextual information such as time, location, contents, features and tags [[61](#_ENREF_61), [62](#_ENREF_62)]. Regarding to how recommendations are made, RS algorithms are categorized into three groups: *content-based*, *collaborative filtering* and *hybrid* recommendation algorithms [[63-65](#_ENREF_63)].

*Content-based recommendation algorithms:* The items recommended using these algorithms have similar contents or features to items that the target user had liked before [[38](#_ENREF_38), [66-70](#_ENREF_66)]. For example, in recommending a movie, the features may include genre, actors, actresses and director. When a user indicates his/her preference in action movies, a content-based algorithm recommends the best-matching action movies to her/him. Moreover, content-based recommenders can exploit the information in users’ profiles [[67](#_ENREF_67)] such as demographic information, age, gender, nationality, education or occupation [[71](#_ENREF_71), [72](#_ENREF_72)] to improve the quality of recommendation.

*Collaborative filtering recommendation algorithms:* This approach recommends the items based on the similarity between the users and/or the items [[17](#_ENREF_17), [63](#_ENREF_63), [65](#_ENREF_65), [73-75](#_ENREF_73)]. The term “collaborative” (first used by the creators of Tapestry recommender [[17](#_ENREF_17)]) refers to the need of collaboration of different users for “filtering” abundant data and generating recommendations. For example, if users *u* and *p* have rated common items similarly in the past, their history of preferences will impact the recommendations each one gets, more than preferences of other non-similar users to them. The algorithms in this category are divided into two subgroups of memory-based and model-based Collaborative Filtering (RS) recommenders [[75-77](#_ENREF_75)], which is discussed in the following sections in more details. This approach is the most widely used RS algorithm in many industrial applications [[58](#_ENREF_58), [64](#_ENREF_64), [65](#_ENREF_65), [76](#_ENREF_76), [78-81](#_ENREF_78)].

*Hybrid recommendation algorithms:* Different combinations of content-based and CF algorithms exist that exploit users’ and items’ information as well as the ratings and similarity of different users and items [[4](#_ENREF_4), [28](#_ENREF_28), [82-91](#_ENREF_82)]. Generally, there are four different ways to combine the content-based and CF recommenders [[63](#_ENREF_63)], including (1) separately implement content-based and CF algorithms, then combine their results [[41](#_ENREF_41), [72](#_ENREF_72), [92](#_ENREF_92), [93](#_ENREF_93)], (2) use some of content-based features to boost a CF algorithm [[4](#_ENREF_4), [72](#_ENREF_72), [84](#_ENREF_84), [94](#_ENREF_94)], (3) or alternatively, use some of the characteristics of a CF algorithm to boost a content-based recommender [[95](#_ENREF_95)], (4) and finally, unify content-based and CF characteristics into one recommender [[82](#_ENREF_82), [83](#_ENREF_83), [96-98](#_ENREF_96)]. Several studies have showed that hybrid recommendation algorithms provide better recommendations than separate content-based or CF algorithms [[4](#_ENREF_4), [72](#_ENREF_72), [84](#_ENREF_84), [95](#_ENREF_95)].

In spite of the improvement and growing research in the field of RS over the past years, there are still several challenges for both content-based and CF methods. Generally, comparing content-based and CF approaches, the latter is not limited to the content or features since they use the users’ rating history for prediction and they can recommend diverse items. However *new user problem* is still an issue, plus other limitations including *new item problem* and *sparsity* [[4](#_ENREF_4), [63](#_ENREF_63), [99](#_ENREF_99)]. The first two problems are usually referred to as “cold-start problem” [[83](#_ENREF_83), [100](#_ENREF_100)]. Combining content-based and CF algorithms – a hybrid approach – has been used as a solution for new user problem in different studies. Other studies has used various methods to find the best item for recommending to a new user and learn her/his preferences gradually; e.g., a simple way is to recommend popular items or items based on the demographic information of user such as age, gender, location, nationality, education or occupation (assuming new users have to fill out a profile before using the system) [[101](#_ENREF_101), [102](#_ENREF_102)].

As the rating matrix grows, the proportion of unrated user-item pairs to rated ones increases, which makes the matrix highly sparse. As sparsity grows, users with unique tastes get poor recommendations, computation time increases, items rated by small number of users are not recommended and generally the quality of RS degrades [[4](#_ENREF_4), [63](#_ENREF_63)]. Various studies addressed the sparsity issue from different perspectives. Pazzani et. al. [[72](#_ENREF_72)] incorporated demographic information of users while calculating users’ similarities. Huang et. al. [[103](#_ENREF_103)] added transitive associations among users when computing users’ similarities, and Billsus et. al. [[104](#_ENREF_104)] and Sarwar et. al. [[105](#_ENREF_105)] used Singular Value Decomposition (SVD) as a dimensionality reduction method for rating matrix. Another issue is how to increase the flexibility of recommendations by generating suggestions in real-time [[63](#_ENREF_63), [106](#_ENREF_106)]. The other issue is scalability of the algorithms due to large number of items and users in these systems which cause computational cost overload. Several solutions has been proposed, such as dimensional reduction approaches, parallelized or incremental algorithms [[46](#_ENREF_46), [55](#_ENREF_55), [79](#_ENREF_79), [107-110](#_ENREF_107)]. Moreover, challenges as how to understand the short and long-term preferences of users and the value of old ratings [[111](#_ENREF_111), [112](#_ENREF_112)], the effect of network structure and how to exploit this additional information to improve recommenders [[113](#_ENREF_113)], and finally detecting different user behaviors, such as new or old users [[114](#_ENREF_114), [115](#_ENREF_115)] are among RS issues..

The quality of RS can be enhanced by various approaches; for example, (1) better knowing users and items [[4](#_ENREF_4), [106](#_ENREF_106), [116](#_ENREF_116)] by incorporating features of their profiles such as demographic information or keywords [[38](#_ENREF_38), [67](#_ENREF_67), [72](#_ENREF_72), [82](#_ENREF_82)] or using data mining techniques [[117-119](#_ENREF_117)], (2) utilizing additional contextual information such as time and location [[63](#_ENREF_63), [120-128](#_ENREF_120)], (3) getting feedbacks on recommendations and improving the performance according to them [[12](#_ENREF_12), [83](#_ENREF_83), [129-131](#_ENREF_129)], (4) by using multi-criteria ratings, e.g., instead of giving a single rating to a movie, users can rate the screenplay, actors, music separately. An example for multi-criteria RS is Zagat’s guide as a restaurant finder with three criteria for rating, namely food, décor and service [[132](#_ENREF_132)].

# 3. Collaborative Filtering Recommendation Algorithms

As mentioned above, CF considers item- or user-based similarities and extract the list of recommendations based on them. To formalize this definition, let us denote “*U*” as the set of users in the system, displayed by and “*I*” as the set of items, represented by. represents the rating that user *u* has given to item *i*, and usually has a numerical scale, such as the five-star rating scale for Movielens[[3]](#footnote-3) (1 means very bad and 5 very good) and the ten-star rating scale for IMDB[[4]](#footnote-4). In some cases, the scale is continuous, e.g.,  for Jester Joke recommender. There are explicit or implicit ways for gathering the users’ ratings, as the indicator of their preference toward the items [[133-135](#_ENREF_133)]. The numerical ratings along binary (like/dislike) and unary ones (e.g., the only choice of *Like* in Facebook), which are entered directly by a user, are considered as explicit ratings [[74](#_ENREF_74)]. We indicate the case when a user has not yet given a rating score for an item by = 0. Furthermore, it is possible to track the users’ behaviors such as mouse movement, browsing time, purchase history, watching or listening time (e.g., whether the user skips the song), frequency of consuming (e.g., a music track), click through data, downloaded applications and similar behaviors as implicit ratings. Implicit feedback gathered from the users can be used to improve the quality of recommendations [[31](#_ENREF_31), [40](#_ENREF_40), [74](#_ENREF_74), [76](#_ENREF_76), [129](#_ENREF_129), [136](#_ENREF_136), [137](#_ENREF_137)]. The data file containing ratings (called user-item interaction matrix) is used for learning the users’ preferences and habits, predicting new ratings, recommending items to users, and finally evaluating the system. For evaluation purposes, we should divide the rating matrix into two parts: training set for learning and test set for evaluating the performance. Memory-based and model-based CF algorithms use these matrices differently, which are discussed in the following subsections.

## 3.1. Memory-based collaborative filtering

As mentioned, memory-based algorithms predict new ratings based on the available data (which is loaded into the memory), using similarity of other users or items to the target user/item [[1](#_ENREF_1), [2](#_ENREF_2), [63](#_ENREF_63), [64](#_ENREF_64), [75](#_ENREF_75), [76](#_ENREF_76), [81](#_ENREF_81), [138](#_ENREF_138), [139](#_ENREF_139)]. The set of similar users to a target user (or similar items to a target item) is called his/its neighbor set and is used to extract users/items with similar history of ratings. The underlying assumption is if two users have similar history of ratings for common items, they will likely have similar preferences for the rest of items. As for the two items rated similarly by several users, they would probably be rated in the same manner by the rest of users. Of course, there are always people with unique tastes and preferences which would not help this case, but generally speaking, this assumption has proved to be useful. After forming the neighborhood, a new rating for a target user-item pair is predicted as a function of the neighbors’ ratings for that particular item and the degree of their similarities to the target user. Based on using the target users/items neighbors, these algorithms are divided into two categories of user-based and item-based CF.

There are various similarity measures for extracting the set of neighbors, such as cosine–based similarity adapted from information retrieval [[140](#_ENREF_140)], adjusted cosine, Pearson correlation coefficient, constrained Pearson correlation, Euclidean and mean squared differences [[64](#_ENREF_64)]. In addition to the similarity, in some cases, one can use dissimilarity values as well [[141](#_ENREF_141)], e.g., when the sparsity of available data is high and the relevance becomes more important than the correlation [[142-144](#_ENREF_142)]. Next, we review cosine-based similarity and Pearson correlation; however as Pearson correlation coefficient has shown to be the more effective technique [[63](#_ENREF_63)], we chose Pearson as our similarity measure in the implementations. Cosine-based similarity between two users *u* and *ú* is calculated using [[20](#_ENREF_20), [76](#_ENREF_76)]



where *Iuú* is the set of items rated by both users, and is the rating that user *u* has given to item *i.* Pearson correlation coefficient for users *u* and *ú* is obtained using [[1](#_ENREF_1), [2](#_ENREF_2)].



where is the average rating of user *u*.

As the similarity values are computed, one can use *k Nearest Neighbor* (KNN) algorithm to find the *k*-most similar users to the target user. We used *k* = 30 as suggested by Sarwar et al. [[20](#_ENREF_20)], in our implementations. Equations and are also applicable for calculating the similarities between items [[20](#_ENREF_20), [145](#_ENREF_145)]. Usually, in real recommenders the similarities are pre-calculated and used whenever there is a need to generate recommendations. This enables to make the recommendation on a real-time fashion. However, the similarity values must be recalculated once in a while, due to the change in the users and items networks in the system [[63](#_ENREF_63)].

### 3.1.1. User-Based Collaborative Filtering

User-based CF is among the most successful and widely implemented techniques in RS [[1](#_ENREF_1), [12](#_ENREF_12), [146](#_ENREF_146)]. It recommends items to a target user based on opinions of other similar users to him/her [[76](#_ENREF_76)]. After forming the neighborhood, the new rating for the target user-item pair is estimated considering the weights of different neighbors. That is, the higher is the similarity of a user with the target user, the more impact his/her rating have on the estimation of the target user’s rating. The new rating for user *u* and item *i* is predicted as using



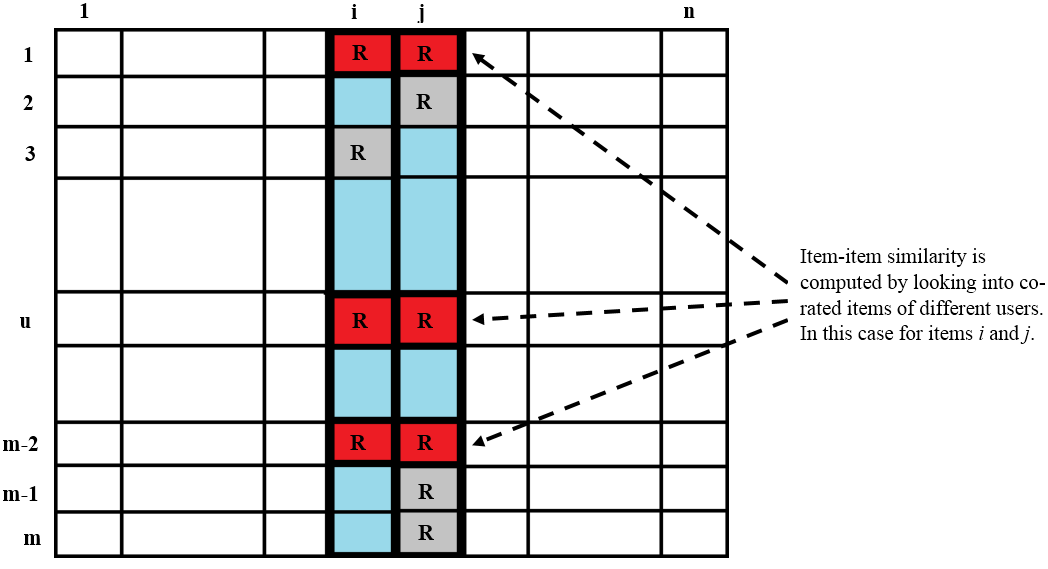
where is the neighbor set of target user *u* with *k* members. However, equation neglects the fact that different people may differently use the rating scale. Equation considers this issue through adjusting the formula by first subtracting the average rating of each neighbor user, , then multiplying the result by his/her similarity with the target user, and finally adding the target user’s average rating [[63](#_ENREF_63), [74](#_ENREF_74)].



User-based CF is a popular method for being relatively fast with reasonable accuracy in the prediction task. However, it has its own drawbacks, such as dealing with sparsity and scalability issues. As the number of items grows, even an active user does not visit a high percentage of total items, and thus the data will be extremely sparse. A possible solution for the sparsity problem is to use semi-intelligent content-boosted agents to increase the density of rating matrix [[94](#_ENREF_94), [147](#_ENREF_147)]. Another solution is to use latent semantic indexing to detect the similarity between users and items in a reduced dimensional space [[54](#_ENREF_54), [105](#_ENREF_105)]. As for the scalability, the computation complexity increases as the number of users and items in the system grow [[20](#_ENREF_20)]. In the worst case, for a user-based CF with *m* users and *n* items, the phase of calculating similarities has a time complexity in order of , and the prediction phase takes the order of . The user-based CF provides low coverage of available items in the system, and the final recommendations are more biased to popular items.

### 3.1.2. Item-Based Collaborative Filtering

In order to alleviate the scalability issue of user-based CF, item-based CF algorithm was proposed [[20](#_ENREF_20)]. The underlying assumption here is that two items which are being rated similarly by several users, would probably be rated in the same manner by the rest of users. Unlike the user-based CF, the item-based algorithm analyzes the rating matrix to identify the relationships between *items*. Prediction is computed by taking a weighted average of the target user's ratings on similar items. Item-based CF is preferred to user-based whenever the number of items in the system is far less than the number of users. Fig. 1, retrievd from Sarwar et al. [[20](#_ENREF_20)], depicts the similarity computation of item-based CF.



**Fig. 1**: Co-rated items and similarity computation in item-based CF. Item-item similairy is computed using ratings by users who rated both items *i* and *j*. The figure and concept is adopted from [[20](#_ENREF_20)].

When estimating an unknown rating value, first Eq. is modified to calculate the Pearson correlation between *items* rather than users. Then, the new ratings are calculated using



where is the neighborhood set of item *i*. According to [[20](#_ENREF_20), [145](#_ENREF_145)], item-based CF is scalable for larger data sets since the item-neighborhood is rather static, and therefore, the similarities can be efficiently pre-computed. Thus, the online (prediction phase) performance is better than user-based CF. Regarding the computational complexity of item-based CF with *m* users and *n* items, the similarity calculation phase needs a time complexity in order of and prediction phase takes . However, in a real system with large number of users and items, the actual complexity is far less, since the number of items not being rated by a user is much less than the total number of items.

### 3.1.3. Resource Allocation Collaborative Filtering

Resource allocation CF uses the notion of link prediction [[148](#_ENREF_148)] in RS to improve its performance. There are a number of methods for link prediction in complex networks [[149](#_ENREF_149)]. Javari et. al. [[150](#_ENREF_150)] used resource allocation (RA) index [[151](#_ENREF_151)], which has shown good estimation of the missing links in bipartite networks of users and items. They used the concept that more popular items should have less impact when calculating the similarities of users, since most users like popular items, and therefore, they are less worthy to be used for the recommendation purposes. RA is a degree-based local similarity measure, in which its value for two users depends on the degree of the common-rated items by users; that is the more popular the common-rated items, the less the value of their RA similarity index. Moreover, as the number of common-rated items grows, RA increases, and the calculated similarity is more trust-worthy. According to this algorithm, if each node distributes its resources equally to all of its neighbors, the RA index, as a degree indicator of the similarity between users *u* and *ú*, is obtained as



where *di* is degree of item *i*, that is the number of users who have rated *i*, and is the set of items rated by *u*. Eq. indicates that the RA values depend on both the number of common neighbors and their degree. As the RA scores are obtained, the similarity between the users is obtained as



The rest is the same as standard user-based CF algorithm.

### 3.1.4. More on memory-based collaborative filtering algorithms

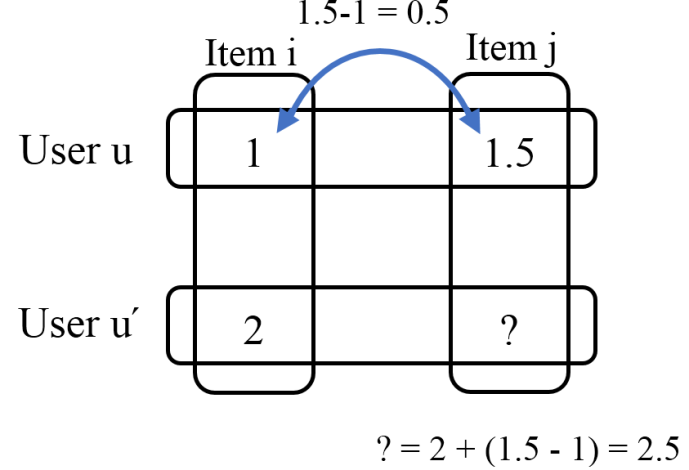
Due to the simplicity of memory-based algorithms, they have been implemented in many industrial recommenders. The KNN technique used in these algorithms is simple and produces reasonably accurate results, and the new data can be added incrementally to the algorithm. Memory-based CF algorithms do not have a learning step (no model is extracted from the data), but their prediction phase may need heavy computations. Despite their wide-spread use in recent years, memory-based CF algorithms have several drawbacks related to synonymy, sparsity and scalability. Indeed, the similarity-based methods cannot be pre-computed for real-time performance [[13](#_ENREF_13)]. According to [[76](#_ENREF_76)], in order to improve the neighborhood-based results, one can utilize several extensions such as default votes, inverse user frequency or case amplification. The first extension assumes a default vote for the pairs of user-item, for which there is no explicit rating. This is performed in order to increase the common items rated by two users (or common users who have rated two items). The idea behind using inverse user frequency is to decrease the effect of universally-liked items when calculating the similarities. The case amplification technique uses a weight transform which would put more emphasis on higher weights, while punishing low weights. Furthermore, as the number of users and items increases, the time complexity of the nearest neighbor algorithms linearly grows [[152](#_ENREF_152)]. A solution can be to use clustering techniques on users or items, in order to reduce the search time [[15](#_ENREF_15), [153](#_ENREF_153)]. Another approach is to use model-based techniques such as dimensionality reduction to deal with these limitations. Several Model-based algorithms are reviewed in the following section.

## 3.2. Model-based collaborative filtering

Model-based CF algorithms use different techniques on the training set, in order to find patterns in the data and learn a model for predicting new ratings [[46](#_ENREF_46), [76](#_ENREF_76), [104](#_ENREF_104), [153-158](#_ENREF_153)]. One can name Slope one [[159](#_ENREF_159)], latent factor models such as Matrix Factorization (MF) and S[ingular Value Decomposition](http://en.wikipedia.org/wiki/Singular_value_decomposition) (SVD) [[104](#_ENREF_104), [105](#_ENREF_105), [110](#_ENREF_110), [160-164](#_ENREF_160)], Bayesian classifiers [[165-168](#_ENREF_165)], [clustering models](http://en.wikipedia.org/wiki/Cluster_Analysis) [[55](#_ENREF_55), [153](#_ENREF_153), [169-171](#_ENREF_169)], various probabilistic relational models [[101](#_ENREF_101), [154](#_ENREF_154)], [probabilistic latent semantic analysis](http://en.wikipedia.org/wiki/Probabilistic_latent_semantic_analysis) [[96](#_ENREF_96), [155](#_ENREF_155), [172-174](#_ENREF_172)], linear regression [[20](#_ENREF_20), [175](#_ENREF_175)], maximum entropy model [[157](#_ENREF_157)], [Latent Dirichlet Allocation](http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation) (LDA) [[156](#_ENREF_156)], [Markov](http://en.wikipedia.org/wiki/Markov_decision_process) chain based models [[176](#_ENREF_176)], principal component analysis (PCA) [[177](#_ENREF_177)], probabilistic factor analysis [[178](#_ENREF_178)], neural networks and fuzzy systems [[34](#_ENREF_34), [35](#_ENREF_35), [60](#_ENREF_60), [179-182](#_ENREF_179)] among the techniques used for model-based CF. In the following, we review a number of frequently used model-based methods. Although there is a rich literature on model-based CF, these algorithms have limited applicability in real scenarios.

### 3.2.1. Slope one

Lemire and Maclachlan proposed three slope one algorithms that pre-compute the average difference between the ratings of two items for users who have rated both items, in the form of predictors, where *b* is a constant and variable *x* represents the ratings [[159](#_ENREF_159)]. For each pair of items, slope one calculates how much an item is more preferred than the other one (*popularity differential*), which is then used to predict the user’s rating of one of those items, given their rating of the other. Note that the distinction between slope one algorithms is due to the way that the relevant differences are selected for the prediction. For example, consider two users *u* and *ú*, and two items *i* and *j*, as shown in Fig. 2. The ratings of user *u* on items *i* and *j* are 1 and 1.5, respectively. User *ú* gave a rating of 2 to item *i* and did not rate item *j*. slope one tries to predict the rating that user *ú* would give to item *j*, using both ratings of user *u* and the rating of a common item between users *u* and *ú* (i.e., item *i*). Since *u* rated *j* 0.5 point (1.5 - 1) higher than *i*, one can predict *ú* will give *j* a rating of 2.5 (2 + 0.5). As described by this example, slope one utilizes the information of other users who have rated the same item, other items rated by the same user and the data points that fall neither in the user vector nor in the item vector (e.g., rating of user *u* to item *i*) when predicting an unknown rating.



**Fig. 2**: Basis of slope one algorithm; adopted from [[159](#_ENREF_159)].

Given two rating vectors *vi* and *wi*, , slope one searches for the best predictor of the form to predict *w* from *v* by minimizing . Therefore, *b* must be chosen to be the average difference between the two vectors. The average deviation of item *i* with respect to item *j* is calculated as



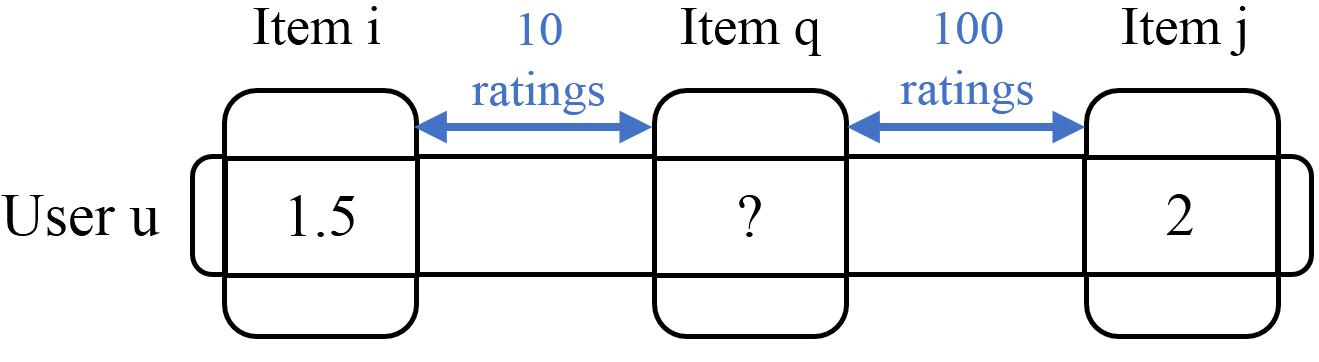
where is the set of users who have rated both items *i* and *j* in the training set, and are the ratings given by user *u* to *i* and *j*, respectively, and *card*(*S*) is the number of elements in set *S*. Note that the symmetric matrix defined by can be computed once and updated quickly when new data is entered. Afterward, unseen ratings are predicted as



where is the set of all relevant items and *S*(*u*) is the set of items rated by user *u*.

### 3.2.2. Weighted slope one

One may notice that in the computations of slope one, the number of observed ratings is not included. Considering the scenario shown in Fig. 3, where *u* has rated both *j* and *i*, we want to predict the utility of item *q* for user *u*. If 100 users had rated both *q* and *j*, and only 10 users had rated both *q* and *i*, the rating given by *u* to *j* is probably a better predictor for *q* (i.e., more reliable) than the rating to *i*.



**Fig. 3**: slope one weighted regarding the number of common ratings between items *i* and *q* and also items *j* and *q*.

The weighted ratings are predicted using [[159](#_ENREF_159)]



### 3.2.3. Matrix Factorization Methods

Due to the limitations of memory-based CF algorithms regarding scalability and sparsity issues, new algorithms tried to address these issues using dimensionality reduction techniques [[105](#_ENREF_105), [183](#_ENREF_183)]. MF methods such as SVD are among these techniques [[164](#_ENREF_164), [184-191](#_ENREF_184)]. In their simplest form, these algorithms factorize the rating matrix into two low-rank matrices: users profile and items profile. High similarity between item and user profiles results in a recommendation. According to [[190](#_ENREF_190)], these methods have several advantages such as better accuracy (in regard to kNN-based algorithms), good scalability, and relatively easy learning process. However, their main burden is the difficulty in learning the model. In the following, we give details on a number of methods based on matrix factorization.

#### 3.2.3.1 Regularized Singular Value Decomposition

In RS algorithms based on SVD, the SVD technique is used to first detect the latent relationships between the users and the items for prediction of unknown ratings, and then to generate a low-dimensional representation of the original rating matrix space to calculate the neighborhood in the reduced space [[105](#_ENREF_105)]. SVD factorizes the rating matrix into a product of two low-rank matrices. It produces a set of uncorrelated eigenvectors which represents the users and the items. For a rating matrix *M* with dimensions and rank *r*, SVD is calculated as



where *m* and *n* are the total number of users and items respectively, and dimensions of *U*, *S* and *V* are , and , respectively. *U* and *V* are two orthogonal matrices, *S* is a diagonal matrix which is called the “singular matrix” and have *r* nonzero diagonal elements. Note that the dimensions of matrices *U* and *V* are reduced to and , respectively and the values on the diameter of *S* are sorted decreasingly. The first *r* columns of *U* and *V* represent the orthogonal eigenvectors associated with the *r* nonzero eigenvalues of *MMT* and *MTM*, respectively. *U* and *V* are called the *left* and *right* singular vectors, respectively [[104](#_ENREF_104), [105](#_ENREF_105)]. One can keep only *k* of *r* singular values (highest values) and discard lower entries. (*r − k*) columns from *U* and (*r − k*) rows from *VT* are eliminated to produce *Uk* and *VTk* matrices. *Uk* and *Vk*are multiplied together using *Sk* to produce *Mk*. The reconstructed matrix *Mk* is the closest rank-*k* matrix to *M*, with respect to the Frobenius norm of matrix. Indeed,



Now, the rating prediction for user *u* and item *i*, , is calculated using dot product as



SVD has several advantages (e.g., predictions with good accuracy), and it addresses the synonymy problem by helping users who have rated similar, but not exact items to be mapped into the space spanned by the same eigenvectors. Furthermore, the low-rankapproximation of the original space is better than the original space itself due to eliminating small singular values which cause noise in the user-item relationship [[110](#_ENREF_110), [192](#_ENREF_192), [193](#_ENREF_193)]. According to [[54](#_ENREF_54), [105](#_ENREF_105), [110](#_ENREF_110)], algorithms based on SVD can make the neighborhood formation process of CF systems very scalable, often resulting in better performance. The space storage of SVD takes , since it only stores two matrices of size and . Therefore, the space storage of SVD is very efficient compared to neighborhood-based CF algorithms that is . Despite its popularity and being implemented in several studies [[194](#_ENREF_194), [195](#_ENREF_195)], SVD is slow and requires dealing with the missing values. Furthermore, its optimization method is non-convex. A solution to deal with the missing values is to replace them with the users’ average ratings or those of the items [[105](#_ENREF_105)]. A typical model-based RS algorithm has two steps: one for model-building (offline) and one for execution (online). A major defect of SVD is its time-consuming offline decomposition step. For an rating matrix, the required time of this step is the order of [[192](#_ENREF_192), [193](#_ENREF_193)], while the complexity of online phase in neighborhood-based CF is , much lower than SVD [[105](#_ENREF_105)]. Incremental SVD was proposed to help improve the computation time [[110](#_ENREF_110)], which first computes a model with an appropriate size, and then use a projection method to build incrementally upon that [[110](#_ENREF_110)].

During the Netflix competition in December 2006, regularized SVD (RegSVD) was proposed by Brandyn Webb (also called FunkSVD) to improve SVD using a learning rate, regularization constants and a method for clipping predictions [[160](#_ENREF_160)]. This algorithm minimizes the squared error between the actual ratings and predicted estimations for all available votes [[196](#_ENREF_196)]. For minimization process, RegSVD uses gradient descent that can achieve good accuracy by choosing appropriate parameters. To handle overfitting issue, one can add regularization terms both for the users and the items profile. Paterek improved this method by introducing biases for each user and item to the RegSVD in his paper, an algorithm they named *RSVD2* [[197](#_ENREF_197)].

#### 3.2.3.2. Non-Negative Matrix Factorization

In the Non-negative MF (NMF) algorithm, there is a low-dimensional linear model with a non-negative constraint to indicate the rating matrix. This means that the profile of the users and items in a NMF, should have only positive values [[184](#_ENREF_184), [198](#_ENREF_198), [199](#_ENREF_199)]. This method uses multiplicative update rules for minimizing the least squares error between the actual ratings and the predicted ones.

#### 3.2.3.3. Probabilistic Matrix Factorization

Probabilistic MF (PMF) utilizes a probabilistic linear model to represent the latent features of users and items. According to Salakhutdinov and Mnih [[185](#_ENREF_185)], implementation of this algorithm on the large and sparse dataset of Netflix showed promising results. Theis model linearly scales with the number of ratings.

Table 1: summarization of recommender algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Recommendation Approaches** | | **Algorithms** | | **Applied Techniques** |
| Content-based RS | Heuristic-based CBF | *- K* Nearest Neighbor  *-* Clustering | | |
| Model-based CBF | *-* Bayesian  *-* Clustering  *-* Neural networks | | |
| Collaborative Filtering RS | Memory-based CF | *-* User-Based CF  *-* Item-Based CF  *-* Resource Allocation CF | *- K*-Nearest Neighbor  *-* Graph Theory  *-* Decision tree  *-* Web mining  *-* Support vector machine  *-* Bayesian models  *-* Clustering  *-* Association mining | |
| Model-based CF | *-* Slope One  *-* Weighted Slope One  *-* Singular Value Decomposition (SVD)  *-* Non-Negative Matrix Factorization (NMF)  *-* Probabilistic MF  *-* Bayesian PMF  *-* Non-Linear BPMF | *-* Principal Component Analysis (PCA)  *-* Bayesian networks  *-* Clustering  *-* Latent semantic analysis  *-* Neural networks  *-* Linear regression  *-* Probabilistic models  *-* Maximum entropy | |
| **Hybrid RS** | | *-* Feature combination  *-* Recommendation results combination  Other | *-* Bayesian  *-* Clustering  *-* Linear combination  *-* Probabilistic models  *-* Maximum entropy | |

#### 3.2.3.4. Bayesian Probabilistic Matrix Factorization

Usually, low-rank MF algorithms are fitted to the data using a Maximum-a-Posteriori (MAP) estimate of the model parameters, which in the case of inaccurate tuning of regularization parameters, will often result in overfitting. Salakhutdinov and Mnih presented a Bayesian version of PMF with automatic controlling of all model parameters and hyperparameters [[186](#_ENREF_186)]. They used Markov Chain Monte Carlo (MCMC) method to train their model and applied it on the Netflix dataset. Their results indicated better accuracy than PMF method.

#### 3.2.3.5. Non-Linear Probabilistic Matrix Factorization

Non-linear PMF (NPMF) uses Gaussian process latent variable models for recommendation. The model is optimized using stochastic gradient descent method. Lawrence and Urtasun applied this model to EachMovie and Movielens datasets and achieved good results [[189](#_ENREF_189)].

Table 1 represents the summary of CF recommender algorithms mentioned above. Note that these algorithms are only representative for RS algorithms and many extensions to these classical memory- and model-based have been proposed in the literature, some of which were also discussed in this manuscript.

# 4. Evaluation Metrics

There are various metrics in the field of RS for evaluating the performance of different algorithms [[74](#_ENREF_74), [200](#_ENREF_200)]. In this section we review the most common evaluation metrics to assess the performance of RS algorithms. Evaluations can be offline or online [[74](#_ENREF_74), [135](#_ENREF_135)]. In an offline analysis, the dataset is collected and a proportion of ratings are hidden from the recommender algorithm as a test set. Then, the target algorithm uses the rest of data (i.e., the training set) to predict new ratings or rank the unseen items. Afterward, one metric or combination of evaluation metrics is used to measure the quality of recommendations. Offline analysis has the advantage of being quick, while its major drawback is that we cannot measure the true satisfaction of users regarding the recommended items. On the other hand, online evaluations are conducted in a live experiment, observing the users’ behavior, recommending items and measuring their satisfaction using their feedback or tracking their acts such as click-through rate. However, conducting an online evaluation is expensive and often impossible. In this paper, we used offline analysis to evaluate RS algorithms on different existing datasets.

There are different perspectives on evaluation metrics; some are based on the recommendation list itself, such as accuracy, coverage, diversity and novelty, some are based on the system’s or users’ point of views independent of the recommender, namely confidence, robustness, adaptivity, scalability, trust, risk and privacy [[201](#_ENREF_201)]. It is logical that different metrics evaluate different aspects of algorithms [[200](#_ENREF_200)], hence, the focus of this paper is to evaluate the algorithms discussed in section 3 using several evaluation metrics including accuracy metrics, rank-based metrics, diversity, novelty and coverage. Each algorithm first produces the predicted ratings, the results are then sorted, and for each user, the top-*N* items with the highest predicted ratings are recommended. The metrics evaluate different properties of these top-*N* items. Note that in this study, in order to have integrity among all datasets, only the ratings of users and the relationships due to these ratings are used, and other data such as timestamps of the ratings are not included.

By a quick review we will find out that a diverse range of metrics have been used by researchers in this field. Among the first studies of RS, Shardanand et. al. [[2](#_ENREF_2)] used *reversal* (errors between the predicated and true ratings) as an evaluation metric. Konstan et. al. [[12](#_ENREF_12)] used *receiver operating characteristic* *curve* (ROC curve) for evaluating RS, which had been used for evaluation in the context of information filtering beforehand. Breese et. al. [[76](#_ENREF_76)] analyzed several CF algorithms and introduced some extensions such as *default vote* and evaluated them using *Mean Absolute Error* (MAE) and *half-life utility*. Several researchers have used accuracy metrics such as MAE and *Root Mean Squared Error* (RMSE) to evaluate RS, while some others used non-accuracy metrics. Mobasher et. al. [[202](#_ENREF_202)] used *coverage* as a measure of quality of recommenders that is defined as the proportion of items that recommender can suggest to users. McNee et. al. [[203](#_ENREF_203)] measured the degree to which recommendations where *surprising* or *non-obvious*. Others have measured the *explainability* of a recommender that is how well a recommender can explains its recommendations to users. A few studies have discussed that these metrics do not measure the users’ satisfaction of the recommender, while others have argued that the users’ satisfaction may not be the ultimate goal of using a recommender in some cases. In the following, we give a comprehensive overview of the metrics that can be used to evaluate the performance of CF methods where the only available information is the rating histories. We categorized these metrics into three groups: *accuracy* metrics, *ranking-based* metrics, and *diversity, novelty and coverage*.

## 4.1 Accuracy metrics

Herlocker et. al. [[74](#_ENREF_74)] categorized accuracy metrics into three classes, namely, *predictive accuracy* metrics, *rank accuracy* metrics and *classification accuracy* metrics, which are reviewed respectively.

### 4.1.1 Predictive Accuracy metrics

In order to measure the accuracy of the predicted ratings generated by a recommender according to the real ratings (i.e., the rating in the test set) of users, predictive accuracy measures such as MAE or RMSE are frequently used in RS. These metrics are mostly useful where the predicted ratings are shown to the users of system [[74](#_ENREF_74)]. Although these metrics are easily deployable due to their simplicity, the predictive accuracy metrics consider the ratings’ space to be uniform, which is not the case in real systems [[200](#_ENREF_200)]. Moreover, these metrics treat all ratings the same, regardless of their position in the recommendation list [[133](#_ENREF_133)], i.e., a one-star error for an item on the top of the recommendation list penalizes the system the same as a one-star error for an item at the end of the list, which probably will not be recommended to the user ever. Thus, these metrics are not suitable in a system with the goal of *finding good items*, which means the users only care about errors of items in the top-*N* list [[74](#_ENREF_74)]. As a solution, one can put more weight on the errors in the top-*N* items of recommendation list than the rest of the prediction list.

#### 4.1.1.1. Mean Absolute Error

MAE is used to measure the average absolute deviation of the predicted ratings from the real ratings of users, as



where represents the predicated rating of the system for user *u* and item *i*. MAE has been used for evaluation of RS in various studies [[2](#_ENREF_2), [73](#_ENREF_73), [76](#_ENREF_76), [146](#_ENREF_146)]. Often, the averaged MAE for all users is shown as a general performance of a RS. In order to compare recommenders using different rating scales, one can normalize the total MAE by dividing the mean MAE value over all users by the maximum rating () minus the minimum rating () in the system [[46](#_ENREF_46)],



#### 4.1.1.2. Root Mean Absolute Error

RMSE is a variation of MAE which puts more emphasis on large errors by using power 2 on the deviation of predicted ratings from real ratings,



and the normalized RMSE as



#### 4.1.1.3. Asymmetric Loss

In systems which recommending bad items as good ones is worse than recommending good items as bad ones, one can use asymmetric loss to evaluate the system



If the recommended item is liked by the user, *loss* equals zero, however for an item is liked by the user but not recommended by the system, *loss* is defined as,



and finally if a disliked item is recommended to the user, the loss is



### 4.1.2 Rank Accuracy metrics

In order to measure the relationship between the order of the items in a recommended list with the order that each user has given to the same items, rank accuracy metrics are used. These metrics are useful when one recommends a ranked list of items to users [[74](#_ENREF_74)].

#### 4.1.2.1. Pearson Correlation

Pearson Correlation (PC) measures the linear relationship between two list of predicted ratings and real ratings of a user. Hill et. al. used this metric to evaluate their recommender algorithm [[3](#_ENREF_3)]. One should first calculate the PC for each user, and then get the average by dividing sum of all PC values by the number of users in the system, as



#### 4.1.2.2. Spearman Correlation

Spearman's rank correlation coefficient or Spearman's rho, is a [nonparametric](http://en.wikipedia.org/wiki/Non-parametric_statistics) metric of [statistical dependence](http://en.wikipedia.org/wiki/Correlation_and_dependence) between predicted and real lists and is named after [Charles Spearman](http://en.wikipedia.org/wiki/Charles_Spearman) [[204](#_ENREF_204)]. The difference between Pearson and Spearman correlations is that in the latter, *ri* and  are the ranks of corresponding items in the real and predicted rating lists, respectively, rather than the value of ratings. Indeed, Spearman’s rank correlation is Pearson correlation of the rank vectors, and is calculated as



#### 4.1.2.3. Kendall Tau Correlation

Kendall rank correlation is used to measure the similarity of the orderings of two ranked lists which was proposed by [Maurice Kendall](http://en.wikipedia.org/wiki/Maurice_Kendall) [[205](#_ENREF_205)]. It is defined as



where *C* is the number of pairs of items for which the recommender predicted in the same order as real rating list of user (Concordant pairs) and *D* is the number of pairs of items for which the recommender predicted the wrong order (Discordant pairs). When there are no discordant pairs of items (*D = 0*), i.e., the ranking of items in the predicted and the real rating lists are exactly the same, the Kendall tau value will be 1, and for two completely dissimilar lists the value of this metric is -1 (*C = 0*). Considering a ranked list with *N* items, there are ordered pairs of items, thus Eq. is equal to



Sometimes a user gives several items the same ratings or a recommender predicts the same ratings for several items. In such cases, we use a variation of Kendall Tau correlation proposed by Herlocker et. al. [[74](#_ENREF_74)] that is defined as



where *TR* is the number of item pairs with the same real ratings and *TP* is the number of items with the same predicted ratings.

The Kendall Tau does not consider the position of the correct ranked items, e.g., between the first and the second rating and between 50th and 51th ratings. One solution for this issue can be to add more weight to concordant pairs at the top of the list than those towards the end of the list [[133](#_ENREF_133)].

#### 4.1.2.4. Normalized Distance-Based Performance Measure

For comparing two weakly ordered ranked list, normalized distance-based performance measure (NDPM) was proposed by Yao [[206](#_ENREF_206)] and is defined by



where *D* is the number of discordant pairs, *CU* is the number of pairs for which one system gives a tie and the other ranked list does not, and *NP* is the total number of pairs in the real ranked list minus tied ones. Indeed, the modified Kendall Tau correlation penalize the system even when there are tied pairs in the real ranked list; however, *NDPM* only penalizes the recommender for tied pairs of predicted ranked list for which one item is strictly preferred in the real list.

### 4.1.3 Classification Accuracy metrics

In order to measure how many times the system can classify a relevant item as a good one or an irrelevant item as a bad one, classification accuracy metric are used. A relevant item is an item that the user has liked in the real ratings. For binary ratings, obviously, there are *liked* and *disliked* items and not the ratings. Usually, the disliked and non-rated items are grouped together, and hence, the liked items (sometimes showed as purchased items) are the relevant ones. However, for numerical rating scales, we define the relevant items as the items for which the user gave a higher rating than the average rating of all the items he/she voted for. For classification metrics, deviation from the real ratings is tolerated as long as the relevant items are recommended on the top-*N* list. Unlike the rank accuracy metrics, these metrics are most useful when evaluating recommenders with binary or unary ratings. In order to apply the classification accuracy metric for datasets with numerical rating scales, for each user we consider items rated more than his/her average rating as items which he/she has liked. Precision, recall and F1 score metrics belong to this group.

One problem with these metrics is that the items with no real rating cannot be considered as irrelevant, because the user might had not seen or consumed them at all [[74](#_ENREF_74)]. In order to handle the sparsity of the input data, there are a number of approaches. One simple solution is to eliminate all unrated items and predict the top-*N* list only for items for which we have the users’ rating. However, this method likely leads to biased recommendation, i.e., the items which the user has not yet consumed will never be measured and get recommended to the user. The second solution for treating sparse datasets is to consider slightly negative default ratings in recommending items that has not been rated [[76](#_ENREF_76)]. However, the default rating may differ greatly from the true rating for unobserved item [[74](#_ENREF_74)]. A third approach, which is employed in this manuscript, is to perform the predictions for all unobserved items, but evaluate in only for top-*N* items in the list.

#### 4.1.3.1. Precision and Recall

Precision and recall are among the most frequently used metrics of information retrieval field introduced by Cleverdon et. al. (1968) [[207](#_ENREF_207)]. They have been among the first series of the metrics used to evaluate RS algorithms [[54](#_ENREF_54), [82](#_ENREF_82), [104](#_ENREF_104), [105](#_ENREF_105)]. These metrics use a confusion matrix that divides the items into 4 different groups (Table 2). In this matrix, relevant items which are recommended by the system, are placed in the true positive (*TP*) group, and those relevant items that the system failed to detect as relevant for the user go to false negative (FN) group. Irrelevant items which are incorrectly recommended by the system are placed in the false positive (FP) group, and finally, the irrelevant items that are correctly not recommended to the user are considered in the true negative (TN) group.

**Table 2**: Confusion matrix. TP represents true positive, TN represents true negative, FP means false positive and FN means false negatives.

|  |  |  |
| --- | --- | --- |
|  | **Recommended** | **Not recommended** |
| **Relevant** | TP | FN |
| **Irrelevant** | FP | TN |

Precision is calculated as the ratio of the relevant items which are recommended to the number of all recommended items, as



And recall is calculated as the ratio of the relevant items which are recommended to the number of all relevant items, as



As mentioned before, precision and recall are not directly applicable to evaluate RS algorithms, and we need to know each item is relevant or not, which means every item must be rated by the user. Thus, we used *P@N* and *R@N* instead (*N* being the size of the recommendation list) defined by [[54](#_ENREF_54)]





where *Relu* is the set of the items relevant to user *u*.

#### 4.1.3.2. F1 Score

Since precision and recall are inversely correlated [[207](#_ENREF_207)], it is needed to consider both of them when evaluating different algorithms. Moreover, and are dependent on the length of the recommendation list. Therefore, researchers have often used F1 score, as a combination of precision and recall, as defines by [[54](#_ENREF_54), [105](#_ENREF_105)].



## 4.2 Rank-based metrics

Instead of comparing the exact value of the predicted ratings with the real ones, rank-based metrics examine the order and position of the items displayed to the user in the recommendation list. *Half-Life Utility, Discounted Cumulative Gain, Rank-Biased Precision and Recovery Rate* are among this group of metrics.

### 4.2.1. Half-Life Utility

Half-life utility metric evaluates the utility of a recommendation list based on a hypothesis that as the rank of the item in the recommendation list decreases, the probability of user’s tendency to examine it reduces exponentially [[76](#_ENREF_76)]. Since users usually tend to pay attention to items at the top of the recommended list, a “half-life” threshold is defined as the rank of the item on the list for which there is a 50-50 chance that user will examine it. Half-life utility is defined by



where *h* is the half-life threshold, *d* is the neutral vote (we set it as the user’s average rating), and represent the rank of item *i* in the recommendation list. For simulations, we set *h* = 5 as suggested by Breese et. al. [[76](#_ENREF_76)]. We calculated *Hu* for each user and obtained the average over all users to compute the overall *Htotal* score; the greater the value of *Htotal*, the better recommender acts according to this metric. This index is calculated as



### 4.2.2. Normalized Discounted Cumulative Gain

Discounted Cumulative Gain (DCG) evaluates the usefulness of an item based on its rank in a recommended list. The more relevant items are with higher ranks, the more valuable the recommendation list is for the user and one becomes more satisfied with the system which saves his/her time. DCG is defined by [[208](#_ENREF_208)]



DCG is calculated for each user regarding his/her real and predicted lists. is the rank of the item in the recommendation list, e.g., for the first item at the top of the list we have *ri* = 1. The lower is the rank of the item in the recommendation list (i.e., toward the end of the list), its share in the cumulated gain becomes less. The discounting function to reduce this share is log-harmonic. Different values of *b*, as the base of the logarithm, control the degree of reduction in items’ shares in DCG; i.e., the greater the values of *b*, the slower the shares decrease. For the item at rank 1 and smaller than *b*, it is logical to use the first part of Eq. , rather than using the logarithmic discount function. According to [[208](#_ENREF_208)], we set *b* = 2 for the experiments. indicates the relevancy of item *i* (with rank ) in the recommendation list; as mentioned before we consider an item relevant to a user, if he/she has rated it more than the average rating of the total items he/she voted. If the item is relevant, we have *Ri* = 1, otherwise *Ri* = 0. For comparison purposes, we need to eliminate the effect of different sizes of the recommendation lists in the metric. The computed DCG value is normalized through dividing by the maximum possible gain (i.e., the perfect ranking according to user’s preference), which is the exact order of items given by the user in real rating list:



We calculated *NDCGu* for each user and get the average over all users, to compute the overall *NDCGtotal* score for the dataset, as



This score will be between 0 and 1. The greater is the value of *NDCGtotal*, the better the recommender works for relevant items regarding their ranks in the recommendation list.

### 4.2.3. Rank-Biased Precision

Similar to DCG, the Rank-biased Precision (RBP) metric attempts to evaluate the recommendation list through giving more shares to highly ranked relevant items for each user. The difference here is the discounting function which is a geometric sequence instead of the logarithmic one in DCG [[209](#_ENREF_209)]. The underlying assumption here is that users often examine the first item of the recommended list, and then with probability *p*, they may check the next item, i.e., there is (*1 - p*) chance that they will not choose the next item. For example, considering *p* = 0.8, the user will check the first item, then examines the second item with probability of 0.8, the third item with probability of 0.82, and finally the *ith* item with probability of 0.8*i*-1. RBP for each user’s recommended list is defined by



where and *Ri* are defined the same as for DCG. Since , the value of RBP is between 0 and 1; the greater this value, the better the system performs according to RBP. The overall RBP averaged for all users is calculated as



The performance of the recommender depends on the choice of *p* such that for small values of *p*, the user only examines the top-ranked items, while for large values of *p*, it may also examines the items in lower ranks.

### 4.2.4. Recovery Rate

Recovery Rate evaluates the performance based on the correct ranking of the items. The value of Recovery metric for user *u* is define as



where *NR* is the set of relevant items in the real ratings list of user *u*, *ri*indicates the rank of item *i* in the recommendation list, and *Ci* is the number of candidate items to recommend to user *u*. The average Recovery rate for the users is calculated as



## 4.3 Diversity, Novelty and Coverage

So far, we reviewed the accuracy and ranking of recommendations generated by recommenders. However, we need to go further and evaluate them based on other criteria; measures to answer questions such as: what percentage of items in the recommendation lists is new to user; how much they surprise the user; how many of them are popular items; do the algorithm always recommend the same group of items to all users; how much similar are the users’ recommendation lists; and alike. Recently, there has been a shift in the community to design recommenders that not only provide accurate recommendations, but also brings the most satisfaction to the users [[199](#_ENREF_199), [210-217](#_ENREF_210)]. Indeed, in many real applications, the users would like to be recommended diverse and novel items. In this section, we review a number of metrics developed to measure how much a recommender is novel, diverse or covers the items available in the system.

### 4.3.1. Diversity

Diversity is a metric with two concepts: intra-diversity and inter-diversity. Intra-diversity is to measure how different are the items recommended to a user, i.e., the diversity of each recommendation list. To this end, we need to measure the similarities between item pairs in the recommendation lists. This similarity can be obtained from the content information of items or using similarity measures such as cosine or Pearson similarity for item rating vectors [[218](#_ENREF_218)]. The Intra-diversity of *N* items of recommendation list for user *u* can be calculated as



where *s*(*Ii*, *Ij*) is the similarity between items *i* and *j*. Note that the lower is the value of *IntraDiversity*, the more diverse items the system recommends to the user. In order to compute the overall *IntraDiversitytotal*(*N*) for all users of a data set, one can calculate *IntraDiversityu(N)* for each user and then get the average of them, as



The second concept is the inter-diversity, which indicates the extent of difference between the recommendation lists of all users [[219](#_ENREF_219)]. We can measure this notion using Hamming distance between the recommendation lists of users’ *u* and *ú* as



where *Quú* is the similarity between the two users’ recommendation lists; this similarity is defined as the number of common items in both lists. It is worth mentioning that the value of this metric is highly dependent on the number of users and the items. The average metric can be calculated as



### 4.3.2. Self-Information based Novelty

Although we assume that the users would like the items similar to what they have liked in the past, it is also a logical assumption that most users would also want to get recommendations for items they have not yet seen, even the items they won’t expect. If the novel recommendations get users’ attention and they like them, the recommender is successful from this point of view. A number of metrics has been proposed for evaluating RS novelty [[135](#_ENREF_135)] and several researchers have used novelty to evaluate their recommenders [[20](#_ENREF_20), [199](#_ENREF_199), [220](#_ENREF_220), [221](#_ENREF_221)]. In order to define a proper measure for the novelty, let us first define a metric to measure popularity of the recommendations. For more popular items, it is more probable that a user have seen or consumed them, and thus, they are less novel for him/her. One can simply measure the popularity of an item by its degree, resulting in the average popularity score as



where *di* is the degree of item *i*. According to a variation of novelty which uses the notion of popularity, namely Self-Information Based Novelty (SIBN), also known as surprisal or unexpectedness, popular items are less novel. SIBN for item *i*, *SIBNi*, is defined as [[199](#_ENREF_199)],



The average SIBN can be simply obtained by making the average of the above relation over all users. For a user *u*, we can calculate SIBN of all top-*N* items recommended to him/her. We can also use an altered version of this metric, effective novelty, which only considers the novelty of items relevant to each user from the top-*N* recommendation list [[220](#_ENREF_220), [221](#_ENREF_221)]. If we are evaluating a recommender based on the novelty of its recommended items, it is logical to consider only the novelty of those items which actually the user would like and consume. Effective SIBN is defined as



### 4.3.3. Coverage

A good recommender not only recommends likeable items to users, but also covers a wide variety of items which the system has to offer. In its simplest form, coverage indicates that as the percentage of the items a recommender can suggest grow higher, the system is performing better. Note that lower values of coverage implies less diversity [[73](#_ENREF_73), [133](#_ENREF_133)].

### 4.3.3.1. Catalog Coverage

Catalog coverage refers to the percentage of distinct items in the top-*N* recommendation lists of users:



where *n* is the total number of items in the system and *Ir*indicates the total number of distinct items in users’ top-*N* lists. An improvement to this definition is to consider only the items which are relevant to a user.

Table 3: Summarization of evaluation metrics.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Symbol** | **Metric** | **The lower (or higher), the better** |
| 1 | **Accuracy metrics** | NMAE | Normalized Mean Absolute Error | Lower |
| 2 | NRMSE | Normalized Root Mean Square Error | Lower |
| 3 | AL | Asymmetric Loss | Lower |
| 4 | PC | Pearson Correlation | Higher |
| 5 | Τ | Kendall’s Tau Correlation | Higher |
| 6 | SC | Spearman Correlation | Higher |
| 7 | NDPM | Normalized Distance-based Performance Measure | Lower |
| 8 | P | Precision | Higher |
| 9 | R | Recall | Higher |
| 10 | F1 | F1 Score | Higher |
| 11 | **Rank-based metrics** | HLU | Half-Life Utility | Higher |
| 12 | NDCG | Normalized Discounted Cumulative Gain | Higher |
| 13 | RBP | Rank-Biased Precision | Higher |
| 14 | RR | Recovery Rate | Lower |
| 15 | **Diversity, Novelty and coverage** | H | Hamming Distance (Diversity) | Higher |
| 16 | Pop | Popularity | Lower |
| 17 | SIBN | Self-Information Based Novelty | Higher |
| 18 | ESIBN | Effective Self-Information Based Novelty | Higher |
| 19 | C | Catalog Coverage | Higher |
| 20 | EC | Entropy Coverage | Higher |

### 4.3.3.2. Entropy Coverage

In addition to the number of items a recommender covers, the frequency of offering different items may vary; some items are frequently recommended to various users (probably popular ones), and some appears less in the recommendation lists. To consider how many times an item is recommended to the users, one can use a modified version of the coverage presented above. Entropy Coverage, *EC,* is defined



where *pi* is the percentage of the recommendation lists that contains item *i*.

Table 3 summarizes the evaluation metrics discussed above.

# 5. Results & Discussion

In order to compare the performance of various recommendation algorithms regarding the evaluation metrics, we used a machine with Intel(R) Corei7-3632 CPU @ 2.1GHz and 6GB of main memory, and JAVA language on a Unix-based operating system (Ubuntu 12.04). All the parameters affecting the computation time (e.g. neighborhood size, learning rate, and number of iterations) were set equally for the algorithms as explained above. We used 10-fold cross-validation strategy.

## 5.1 Dataset

We used Movielens as one of the most frequently used datasets within the community of RSs. According to the website[[5]](#footnote-5), the datasets were gathered over various periods of time, depending on the size of the set. The items are movies and the rating scale is 1 to 5 stars. We used MovieLens\_100k dataset in this study, in which users with less than 20 ratings were already omitted. It contains 943 users and 1682 items, with rating density of approximately 6 percent.

## 5.2 Performance Evaluation

We implemented the memory-based and model-based CF algorithms introduced in the previous sections, and the results are summarized in Tables 4. For visualizing effect, the values in **boldface** are the best values in the same column, indicating the top-performer RS algorithm for that evaluation metric. It is worth mentioning that in addition to the these evaluation metrics, there are other factors affecting the performance of an algorithm such as time and space complexity, size of the model, density of datasets and more. Therefore, there would not be a *best choice* (i.e., golden algorithm), and one should choose the most useful approach and corresponding metrics according to the recommender’s goals and application, while considering the limitation it would face regarding the data and time available.

The results show significant variability between the algorithms and none of them has the best performance in all evaluation metrics. Often, each research work considers some of these evaluation metrics and assesses the performance of their proposed algorithm on them. However, to have a fare conclusion on the performance, one should consider various evaluation metrics as many as possible. Although it is not easy to compare the algorithm, one can make some conclusions from Table 4. For example, item-based CF often has better performance than user-based CF. However, it has expensive computations to test the algorithm, which is not a burden in real scenarios, in which only the training phase of the algorithm matters. In model-based CF algorithms, PMF shows the best performance being the top-performer in 6 out of 18 evaluation metrics.

# 6. Conclusion

In this work, we compared a number of well-known collaborative filtering (CF) algorithms in terms of various evaluation metrics. We considered various memory- and model-based CF and briefly discussed their advantages and drawbacks. The memory-based CF algorithms are usually faster at query time, while model-based CF algorithms are more scalable and have better performance for sparse data. However, the latter have expensive model building, which increases the time of training step. Therefore, there will be a tradeoff between performance and scalability to address. The evaluation metrics were categorized into four groups: *i*) computation time, *ii*) accuracy-based, *iii*) ranking-based, and *iv*) diversity, novelty and coverage metrics. We compared the RS algorithms in terms of these evaluation metrics on Movielens, which has been frequently used as a benchmark in RS.

# 6. Conclusion

Mahdi Jalili is supported by Australian Research Council under its program on Discovery Early Carrier Research Award (DECRA) Fellowship with project No DE140100620.

Table 4: Performance of CF algorithms on different evaluation metrics - Movielens dataset.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Category** | | **Accuracy-based** | | | | | **Diversity and novelty** | | | | **Coverage** | | **Rank-based** | | | | | | **Time (sec)** | |
| Metrics/algorithms | | NMAE | NRMSE | P@10 | R@10 | F1 | H | Pop | N | EN | C | EC | HLU | NDCG | RBP | τ | SPD | RR | training | testing |
| memory-based approach | User-Based | 0.182 | **0.230** | 0.009 | 0.010 | 0.010 | 0.746 | 0.440 | 8.150 | 0.175 | 0.169 | 0.259 | 0.001 | **0.818** | 0.008 | 0.376 | 0.337 | 0.258 | **0.1** | 118.5 |
| Item-Based | **0.178** | **0.230** | **0.040** | 0.028 | 0.032 | 0.768 | 0.313 | 5.372 | 0.560 | 0.191 | 0.256 | 0.003 | **0.818** | 0.029 | **0.373** | **0.335** | **0.220** | **0.1** | 3333.1 |
| RA | 0.187 | 0.241 | 0.012 | 0.012 | 0.012 | 0.883 | 0.493 | 7.069 | 0.212 | 0.243 | 0.218 | 0.001 | 0.815 | 0.009 | 0.386 | 0.350 | 0.255 | **0.1** | 225.4 |
| model-based approach | Slope 1 | 0.184 | 0.234 | 0.0001 | 0.0001 | 0.0001 | 0.526 | 0.536 | **9.336** | 0.029 | 0.082 | 0.234 | 0.000 | 0.813 | 0.000 | 0.385 | 0.353 | 0.299 | 5.7 | 86.6 |
| Reg-SVD | 0.182 | **0.229** | 0.027 | 0.023 | 0.025 | 0.604 | 0.262 | 7.048 | 0.455 | 0.203 | 0.217 | 0.002 | **0.818** | 0.021 | 0.376 | 0.338 | 0.223 | 39.3 | 72.0 |
| NMF | 0.193 | 0.245 | 0.001 | 0.0001 | 0.0001 | 0.481 | 0.501 | 9.090 | 0.028 | 0.043 | 0.248 | 0.000 | 0.815 | 0.000 | 0.405 | 0.375 | 0.309 | 3.4 | 71.1 |
| PMF | 0.192 | 0.239 | **0.040** | **0.034** | **0.037** | 0.753 | 0.381 | 5.394 | **0.617** | 0.140 | 0.267 | **0.004** | 0.816 | **0.032** | 0.374 | 0.332 | 0.252 | 606.5 | 73.7 |
| BPMF | 0.186 | 0.237 | 0.007 | 0.006 | 0.007 | 0.558 | **0.236** | 6.600 | 0.158 | 0.044 | **0.285** | 0.001 | **0.818** | 0.003 | 0.389 | 0.354 | 0.298 | 1.6 | **70.3** |
| NLPMF | 0.210 | 0.261 | 0.004 | 0.007 | 0.005 | **0.980** | 0.322 | 6.104 | 0.126 | **0.920** | 0.070 | 0.000 | 0.814 | 0.003 | 0.457 | 0.443 | 0.338 | 120.6 | 313.5 |

# References

1. Resnick, P., et al. *GroupLens: an open architecture for collaborative filtering of netnews*. in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994. ACM.

2. Shardanand, U. and P. Maes. *Social information filtering: algorithms for automating “word of mouth”*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995. ACM Press/Addison-Wesley Publishing Co.

3. Hill, W., et al. *Recommending and evaluating choices in a virtual community of use*. in *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995. ACM Press/Addison-Wesley Publishing Co.

4. Balabanović, M. and Y. Shoham, *Fab: content-based, collaborative recommendation.* Communications of the ACM, 1997. **40**(3): p. 66-72.

5. Resnick, P. and H.R. Varian, *Recommender systems.* Communications of the ACM, 1997. **40**(3): p. 56-58.

6. Bennett, J. and S. Lanning. *The netflix prize*. in *Proceedings of KDD cup and workshop*. 2007.

7. Bell, R.M., Y. Koren, and C. Volinsky, *All together now: A perspective on the netflix prize.* Chance, 2010. **23**(1): p. 24-24.

8. Narayanan, A. and V. Shmatikov, *How to break anonymity of the netflix prize dataset.* arXiv preprint cs/0610105, 2006.

9. Narayanan, A. and V. Shmatikov. *Robust de-anonymization of large sparse datasets*. in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. 2008. IEEE.

10. Chen, P.-Y., S.-y. Wu, and J. Yoon, *The impact of online recommendations and consumer feedback on sales.* 2004.

11. Kitts, B., D. Freed, and M. Vrieze. *Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities*. in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000. ACM.

12. Konstan, J.A., et al., *GroupLens: applying collaborative filtering to Usenet news.* Communications of the ACM, 1997. **40**(3): p. 77-87.

13. Schafer, J.B., J.A. Konstan, and J. Riedl, *E-commerce recommendation applications*, in *Applications of Data Mining to Electronic Commerce*. 2001, Springer. p. 115-153.

14. Beeferman, D. and A. Berger. *Agglomerative clustering of a search engine query log*. in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000. ACM.

15. Mobasher, B., et al., *Discovery and evaluation of aggregate usage profiles for web personalization.* Data mining and knowledge discovery, 2002. **6**(1): p. 61-82.

16. Rich, E., *User modeling via stereotypes.* Cognitive science, 1979. **3**(4): p. 329-354.

17. Goldberg, D., et al., *Using collaborative filtering to weave an information tapestry.* Communications of the ACM, 1992. **35**(12): p. 61-70.

18. Miller, B.N., et al. *MovieLens unplugged: experiences with an occasionally connected recommender system*. in *Proceedings of the 8th international conference on Intelligent user interfaces*. 2003. ACM.

19. Miller, B.N., et al., *Movielens unplugged: Experiences with a recommender system on four mobile devices.* People and Computers, 2004: p. 263-280.

20. Sarwar, B., et al. *Item-based collaborative filtering recommendation algorithms*. in *Proceedings of the 10th international conference on World Wide Web*. 2001. ACM.

21. Winoto, P. and T.Y. Tang, *The role of user mood in movie recommendations.* Expert Systems with Applications, 2010. **37**(8): p. 6086-6092.

22. Carrer-Neto, W., et al., *Social knowledge-based recommender system. Application to the movies domain.* Expert Systems with Applications, 2012. **39**(12): p. 10990-11000.

23. Mukherjee, R., N. Sajja, and S. Sen, *A movie recommendation system–an application of voting theory in user modeling.* User Modeling and User-Adapted Interaction, 2003. **13**(1-2): p. 5-33.

24. Lekakos, G. and P. Caravelas, *A hybrid approach for movie recommendation.* Multimedia tools and applications, 2008. **36**(1-2): p. 55-70.

25. Nanopoulos, A., et al., *Musicbox: Personalized music recommendation based on cubic analysis of social tags.* Audio, Speech, and Language Processing, IEEE Transactions on, 2010. **18**(2): p. 407-412.

26. Logan, B. *Music Recommendation from Song Sets*. in *ISMIR*. 2004. Citeseer.

27. Cano, P., M. Koppenberger, and N. Wack. *Content-based music audio recommendation*. in *Proceedings of the 13th annual ACM international conference on Multimedia*. 2005. ACM.

28. Yoshii, K., et al. *Hybrid Collaborative and Content-based Music Recommendation Using Probabilistic Model with Latent User Preferences*. in *ISMIR*. 2006.

29. Park, H.-S., J.-O. Yoo, and S.-B. Cho, *A context-aware music recommendation system using fuzzy bayesian networks with utility theory*, in *Fuzzy systems and knowledge discovery*. 2006, Springer. p. 970-979.

30. Bu, J., et al. *Music recommendation by unified hypergraph: combining social media information and music content*. in *Proceedings of the international conference on Multimedia*. 2010. ACM.

31. Lee, S.K., Y.H. Cho, and S.H. Kim, *Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations.* Information Sciences, 2010. **180**(11): p. 2142-2155.

32. Chen, H.-C. and A.L. Chen. *A music recommendation system based on music data grouping and user interests*. in *CIKM*. 2001.

33. Porcel, C., J.M. Moreno, and E. Herrera-Viedma, *A multi-disciplinar recommender system to advice research resources in university digital libraries.* Expert systems with Applications, 2009. **36**(10): p. 12520-12528.

34. Porcel, C. and E. Herrera-Viedma, *Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries.* Knowledge-Based Systems, 2010. **23**(1): p. 32-39.

35. Serrano-Guerrero, J., et al., *A google wave-based fuzzy recommender system to disseminate information in University Digital Libraries 2.0.* Information Sciences, 2011. **181**(9): p. 1503-1516.

36. Porcel, C., et al., *A hybrid recommender system for the selective dissemination of research resources in a Technology Transfer Office.* Information Sciences, 2012. **184**(1): p. 1-19.

37. Mooney, R.J., P.N. Bennett, and L. Roy. *Book recommending using text categorization with extracted information*. in *Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08*. 1998.

38. Mooney, R.J. and L. Roy. *Content-based book recommending using learning for text categorization*. in *Proceedings of the fifth ACM conference on Digital libraries*. 2000. ACM.

39. Crespo, R.G., et al., *Recommendation System based on user interaction data applied to intelligent electronic books.* Computers in Human Behavior, 2011. **27**(4): p. 1445-1449.

40. Núñez-Valdéz, E.R., et al., *Implicit feedback techniques on recommender systems applied to electronic books.* Computers in Human Behavior, 2012. **28**(4): p. 1186-1193.

41. Claypool, M., et al. *Combining content-based and collaborative filters in an online newspaper*. in *Proceedings of ACM SIGIR workshop on recommender systems*. 1999. Citeseer.

42. Billsus, D. and M.J. Pazzani. *A personal news agent that talks, learns and explains*. in *Proceedings of the third annual conference on Autonomous Agents*. 1999. ACM.

43. Chiang, J.H. and Y.C. Chen, *An intelligent news recommender agent for filtering and categorizing large volumes of text corpus.* International Journal of Intelligent Systems, 2004. **19**(3): p. 201-216.

44. Tintarev, N. and J. Masthoff. *Similarity for news recommender systems*. in *Proceedings of the AH’06 Workshop on Recommender Systems and Intelligent User Interfaces*. 2006. Citeseer.

45. Lee, H.J. and S.J. Park, *MONERS: A news recommender for the mobile web.* Expert Systems with Applications, 2007. **32**(1): p. 143-150.

46. Goldberg, K., et al., *Eigentaste: A constant time collaborative filtering algorithm.* Information Retrieval, 2001. **4**(2): p. 133-151.

47. Zaíane, O.R. *Building a recommender agent for e-learning systems*. in *Computers in Education, 2002. Proceedings. International Conference on*. 2002. IEEE.

48. Jie, L., *A personalized e-learning material recommender system.* 2004.

49. Tang, T. and G. McCalla, *Smart recommendation for an evolving e-learning system: Architecture and experiment.* International Journal on E-learning, 2005. **4**(1): p. 105-129.

50. Soonthornphisaj, N., E. Rojsattarat, and S. Yim-Ngam, *Smart e-learning using recommender system*, in *Computational intelligence*. 2006, Springer. p. 518-523.

51. Khribi, M.K., M. Jemni, and O. Nasraoui. *Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval*. in *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on*. 2008. IEEE.

52. Bobadilla, J., F. Serradilla, and A. Hernando, *Collaborative filtering adapted to recommender systems of e-learning.* Knowledge-Based Systems, 2009. **22**(4): p. 261-265.

53. Schafer, J.B., J. Konstan, and J. Riedi. *Recommender systems in e-commerce*. in *Proceedings of the 1st ACM conference on Electronic commerce*. 1999. ACM.

54. Sarwar, B., et al. *Analysis of recommendation algorithms for e-commerce*. in *Proceedings of the 2nd ACM conference on Electronic commerce*. 2000. ACM.

55. Sarwar, B.M., et al. *Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering*. in *Proceedings of the fifth international conference on computer and information technology*. 2002.

56. Cho, Y.H. and J.K. Kim, *Application of Web usage mining and product taxonomy to collaborative recommendations in e-commerce.* Expert systems with Applications, 2004. **26**(2): p. 233-246.

57. Wen, Y. and Y. Shui-Sheng, *A survey of collaborative filtering algorithm applied in E-commerce recommender system.* Computer Technology and Development, 2006. **16**(9): p. 70-72.

58. Huang, Z., D. Zeng, and H. Chen, *A comparison of collaborative-filtering recommendation algorithms for e-commerce.* Intelligent Systems, IEEE, 2007. **22**(5): p. 68-78.

59. Wei, K., J. Huang, and S. Fu. *A survey of e-commerce recommender systems*. in *Service Systems and Service Management, 2007 International Conference on*. 2007. IEEE.

60. Castro-Schez, J.J., et al., *A highly adaptive recommender system based on fuzzy logic for B2C e-commerce portals.* Expert Systems with Applications, 2011. **38**(3): p. 2441-2454.

61. Bobadilla, J., et al., *Recommender systems survey.* Knowledge-Based Systems, 2013.

62. De Gemmis, M., et al. *Integrating tags in a semantic content-based recommender*. in *Proceedings of the 2008 ACM conference on Recommender systems*. 2008. ACM.

63. Adomavicius, G. and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.* Knowledge and Data Engineering, IEEE Transactions on, 2005. **17**(6): p. 734-749.

64. Candillier, L., F. Meyer, and M. Boullé, *Comparing state-of-the-art collaborative filtering systems*, in *Machine Learning and Data Mining in Pattern Recognition*. 2007, Springer. p. 548-562.

65. Schafer, J.B., et al., *Collaborative filtering recommender systems*, in *The adaptive web*. 2007, Springer. p. 291-324.

66. Lang, K. *Newsweeder: Learning to filter netnews*. in *In Proceedings of the Twelfth International Conference on Machine Learning*. 1995. Citeseer.

67. Pazzani, M. and D. Billsus, *Learning and revising user profiles: The identification of interesting web sites.* Machine learning, 1997. **27**(3): p. 313-331.

68. Debnath, S., N. Ganguly, and P. Mitra. *Feature weighting in content based recommendation system using social network analysis*. in *Proceedings of the 17th international conference on World Wide Web*. 2008. ACM.

69. Van Meteren, R. and M. Van Someren. *Using content-based filtering for recommendation*. in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 2000.

70. Lops, P., M. de Gemmis, and G. Semeraro, *Content-based recommender systems: State of the art and trends*, in *Recommender Systems Handbook*. 2011, Springer. p. 73-105.

71. Krulwich, B., *Lifestyle finder: Intelligent user profiling using large-scale demographic data.* AI magazine, 1997. **18**(2): p. 37.

72. Pazzani, M.J., *A framework for collaborative, content-based and demographic filtering.* Artificial Intelligence Review, 1999. **13**(5-6): p. 393-408.

73. Herlocker, J.L., et al. *An algorithmic framework for performing collaborative filtering*. in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999. ACM.

74. Herlocker, J.L., et al., *Evaluating collaborative filtering recommender systems.* ACM Transactions on Information Systems (TOIS), 2004. **22**(1): p. 5-53.

75. Su, X. and T.M. Khoshgoftaar, *A survey of collaborative filtering techniques.* Advances in artificial intelligence, 2009. **2009**: p. 4.

76. Breese, J.S., D. Heckerman, and C. Kadie. *Empirical analysis of predictive algorithms for collaborative filtering*. in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. 1998. Morgan Kaufmann Publishers Inc.

77. Koren, Y. and R. Bell, *Advances in collaborative filtering*, in *Recommender Systems Handbook*. 2011, Springer. p. 145-186.

78. Marlin, B., *Collaborative filtering: A machine learning perspective*. 2004, University of Toronto.

79. Cacheda, F., et al., *Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems.* ACM Trans. Web, 2011. **5**(1): p. 1-33.

80. Bobadilla, J., et al., *A framework for collaborative filtering recommender systems.* Expert Systems with Applications, 2011. **38**(12): p. 14609-14623.

81. Sun, X., F. Kong, and S. Ye. *A comparison of several algorithms for collaborative filtering in startup stage*. in *Networking, Sensing and Control, 2005. Proceedings. 2005 IEEE*. 2005. IEEE.

82. Basu, C., H. Hirsh, and W. Cohen. *Recommendation as classification: Using social and content-based information in recommendation*. in *AAAI/IAAI*. 1998.

83. Schein, A.I., et al. *Methods and metrics for cold-start recommendations*. in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002. ACM.

84. Melville, P., R.J. Mooney, and R. Nagarajan. *Content-boosted collaborative filtering for improved recommendations*. in *AAAI/IAAI*. 2002.

85. Soboroff, I.M. and C.K. Nicholas, *Related, but not relevant: content-based collaborative filtering in TREC-8.* Information Retrieval, 2002. **5**(2-3): p. 189-208.

86. Burke, R., *Hybrid recommender systems: Survey and experiments.* User modeling and user-adapted interaction, 2002. **12**(4): p. 331-370.

87. Li, Q. and B.M. Kim. *Clustering approach for hybrid recommender system*. in *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*. 2003. IEEE.

88. Ziegler, C.-N., G. Lausen, and L. Schmidt-Thieme. *Taxonomy-driven computation of product recommendations*. in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. 2004. ACM.

89. Kim, B.M. and Q. Li. *Probabilistic model estimation for collaborative filtering based on items attributes*. in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*. 2004. IEEE Computer Society.

90. Pennock, D.M., et al. *Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach*. in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. 2000. Morgan Kaufmann Publishers Inc.

91. Su, X., et al. *Hybrid collaborative filtering algorithms using a mixture of experts*. in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. 2007. IEEE Computer Society.

92. Tran, T. and R. Cohen. *Hybrid recommender systems for electronic commerce*. in *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*. 2000.

93. Billsus, D. and M.J. Pazzani, *User modeling for adaptive news access.* User modeling and user-adapted interaction, 2000. **10**(2-3): p. 147-180.

94. Good, N., et al. *Combining collaborative filtering with personal agents for better recommendations*. in *AAAI/IAAI*. 1999.

95. Soboroff, I. and C. Nicholas. *Combining content and collaboration in text filtering*. in *Proceedings of the IJCAI*. 1999.

96. Popescul, A., D.M. Pennock, and S. Lawrence. *Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments*. in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. 2001. Morgan Kaufmann Publishers Inc.

97. Ansari, A., S. Essegaier, and R. Kohli, *Internet recommendation systems.* Journal of Marketing research, 2000. **37**(3): p. 363-375.

98. Condliff, M.K., et al. *Bayesian mixed-effects models for recommender systems*. in *Proc. ACM SIGIR*. 1999. Citeseer.

99. Lee, W.S. *Collaborative learning for recommender systems*. in *ICML*. 2001.

100. Lam, X.N., et al. *Addressing cold-start problem in recommendation systems*. in *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. 2008. ACM.

101. Yu, K., et al., *Probabilistic memory-based collaborative filtering.* Knowledge and Data Engineering, IEEE Transactions on, 2004. **16**(1): p. 56-69.

102. Rashid, A.M., et al. *Getting to know you: learning new user preferences in recommender systems*. in *Proceedings of the 7th international conference on Intelligent user interfaces*. 2002. ACM.

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

104. Billsus, D. and M.J. Pazzani. *Learning Collaborative Information Filters*. in *ICML*. 1998.

105. Sarwar, B., et al., *Application of dimensionality reduction in recommender system-a case study*. 2000, DTIC Document.

106. Adomavicius, G. and A. Tuzhilin, *Multidimensional recommender systems: a data warehousing approach*, in *Electronic commerce*. 2001, Springer. p. 180-192.

107. Aggarwal, C.C., et al. *Horting hatches an egg: A new graph-theoretic approach to collaborative filtering*. in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999. ACM.

108. Sarwar, B.M. and J.T. Adviser-Riedl, *Sparsity, scalability, and distribution in recommender systems*. 2001: University of Minnesota.

109. Takács, G., et al., *Scalable collaborative filtering approaches for large recommender systems.* The Journal of Machine Learning Research, 2009. **10**: p. 623-656.

110. Sarwar, B., et al. *Incremental singular value decomposition algorithms for highly scalable recommender systems*. in *Fifth International Conference on Computer and Information Science*. 2002. Citeseer.

111. Min, S.-H. and I. Han, *Detection of the customer time-variant pattern for improving recommender systems.* Expert Systems with Applications, 2005. **28**(2): p. 189-199.

112. Xiang, L., et al. *Temporal recommendation on graphs via long-and short-term preference fusion*. in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010. ACM.

113. Huang, Z., D.D. Zeng, and H. Chen, *Analyzing consumer-product graphs: Empirical findings and applications in recommender systems.* Management Science, 2007. **53**(7): p. 1146-1164.

114. Zhang, C.-J. and A. Zeng, *Behavior patterns of online users and the effect on information filtering.* Physica A: Statistical Mechanics and its Applications, 2012. **391**(4): p. 1822-1830.

115. Chen, L. and P. Pu, *Critiquing-based recommenders: survey and emerging trends.* User Modeling and User-Adapted Interaction, 2012. **22**(1-2): p. 125-150.

116. Konstan, J.A., et al. *Recommender systems: A grouplens perspective*. in *Recommender Systems: Papers from the 1998 Workshop (AAAI Technical Report WS-98-08)*. 1998.

117. Adomavicius, G. and A. Tuzhilin, *Expert-driven validation of rule-based user models in personalization applications.* Data Mining and Knowledge Discovery, 2001. **5**(1-2): p. 33-58.

118. Mannila, H., H. Toivonen, and A.I. Verkamo, *Discovery of frequent episodes in event sequences.* Data Mining and Knowledge Discovery, 1997. **1**(3): p. 259-289.

119. Cortes, C., et al. *Hancock: a language for extracting signatures from data streams*. in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000. ACM.

120. Adomavicius, G., et al., *Incorporating contextual information in recommender systems using a multidimensional approach.* ACM Transactions on Information Systems (TOIS), 2005. **23**(1): p. 103-145.

121. Ricci, F., L. Rokach, and B. Shapira, *Introduction to recommender systems handbook*, in *Recommender Systems Handbook*. 2011, Springer. p. 1-35.

122. Adomavicius, G. and A. Tuzhilin, *Context-aware recommender systems*, in *Recommender systems handbook*. 2011, Springer. p. 217-253.

123. Woerndl, W. and J. Schlichter. *Introducing context into recommender systems*. in *Proceedings of AAAI Workshop on Recommender Systems in E-Commerce*. 2007.

124. Woerndl, W. and G. Groh. *Utilizing physical and social context to improve recommender systems*. in *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*. 2007. IEEE Computer Society.

125. Gonzalez, G., et al. *Embedding emotional context in recommender systems*. in *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. 2007. IEEE.

126. Chen, A., *Context-aware collaborative filtering system: Predicting the user’s preference in the ubiquitous computing environment*, in *Location-and Context-Awareness*. 2005, Springer. p. 244-253.

127. Daneshmand, S.H., et al., *A time-aware recommender system based on dependency network of items.* The Computer Journal, 2015. **58**(9): p. 1255-1266.

128. Abbasi, A., et al., *Enhancing precision of Markov-based recommenders using location information*, in *Advances in Computing, Communications and Informatics*. 2014. p. 188-193.

129. Oard, D.W. and J. Kim. *Implicit feedback for recommender systems*. in *Proceedings of the AAAI workshop on recommender systems*. 1998. Wollongong.

130. Caglayan, A., et al., *Learn sesame a learning agent engine.* Applied Artificial Intelligence, 1997. **11**(5): p. 393-412.

131. Middleton, S.E., N.R. Shadbolt, and D.C. De Roure, *Ontological user profiling in recommender systems.* ACM Transactions on Information Systems (TOIS), 2004. **22**(1): p. 54-88.

132. McCarthy, J.F. *Pocket restaurantfinder: A situated recommender system for groups*. in *Proceedings of the Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems, Minneapolis*. 2002.

133. Lü, L., et al., *Recommender systems.* Physics Reports, 2012. **519**(1): p. 1-49.

134. Hernández del Olmo, F. and E. Gaudioso, *Evaluation of recommender systems: A new approach.* Expert Systems with Applications, 2008. **35**(3): p. 790-804.

135. Shani, G. and A. Gunawardana, *Evaluating recommendation systems*, in *Recommender systems handbook*. 2011, Springer. p. 257-297.

136. Choi, K., et al., *A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis.* Electronic Commerce Research and Applications, 2012. **11**(4): p. 309-317.

137. Hu, Y., Y. Koren, and C. Volinsky. *Collaborative filtering for implicit feedback datasets*. in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. 2008. IEEE.

138. Nakamura, A. and N. Abe. *Collaborative Filtering Using Weighted Majority Prediction Algorithms*. in *ICML*. 1998.

139. Delgado, J. and N. Ishii. *Memory-based weighted majority prediction*. in *SIGIR Workshop Recomm. Syst. Citeseer*. 1999. Citeseer.

140. Salton, G. and M.J. McGill, *Introduction to modern information retrieval.* 1986.

141. Zeng, W., et al., *Can Dissimilar Users Contribute to Accuracy and Diversity of Personalized Recommendation?* International Journal of Modern Physics C, 2010. **21**(10): p. 1217-1227.

142. Wang, J., A.P. De Vries, and M.J. Reinders, *Unified relevance models for rating prediction in collaborative filtering.* ACM Transactions on Information Systems (TOIS), 2008. **26**(3): p. 16.

143. Shang, M.-S., et al., *Relevance is more significant than correlation: Information filtering on sparse data.* EPL (Europhysics Letters), 2009. **88**(6): p. 68008.

144. Bobadilla, J., et al., *Collaborative filtering based on significances.* Information Sciences, 2012. **185**(1): p. 1-17.

145. Deshpande, M. and G. Karypis, *Item-based top-n recommendation algorithms.* ACM Transactions on Information Systems (TOIS), 2004. **22**(1): p. 143-177.

146. Miller, B.N., J. Riedl, and J. Konstan. *Experiences with grouplens: Making usenet useful again*. in *Proceedings of the 1997 Usenix Winter Technical Conference*. 1997.

147. Sarwar, B.M., et al. *Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system*. in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 1998. ACM.

148. Liben-Nowelly, D. and J. Kleinberg. *The link prediction problem for social networks*. in *Twelfth Annual ACM International Conference on Information and Knowledge Management*. 2003.

149. Lü, L. and T. Zhou, *Link prediction in complex networks: A survey.* Physica A, 2011. **390**: p. 1150-1170.

150. Javari, A., J. Gharibshah, and M. Jalili, *Recommender systems based on collaborative filtering and resource allocation.* Social Network Analysis and Mining, 2014. **4**(1): p. 234.

151. Tao Zhou, L.L., Yi-Cheng Zhang, *Predicting Missing Links via Local Information.* Europe Physics, 2009. **71**: p. 623-630.

152. Karypis, G. *Evaluation of item-based top-n recommendation algorithms*. in *Proceedings of the tenth international conference on Information and knowledge management*. 2001. ACM.

153. Ungar, L.H. and D.P. Foster. *Clustering methods for collaborative filtering*. in *AAAI Workshop on Recommendation Systems*. 1998.

154. Getoor, L. and M. Sahami. *Using probabilistic relational models for collaborative filtering*. in *Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*. 1999. Citeseer.

155. Hofmann, T. *Collaborative filtering via gaussian probabilistic latent semantic analysis*. in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. 2003. ACM.

156. Marlin, B.M. *Modeling user rating profiles for collaborative filtering*. in *Advances in neural information processing systems*. 2003.

157. Pavlov, D.Y. and D.M. Pennock. *A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains*. in *Advances in Neural Information Processing Systems*. 2002.

158. Yu, K., et al. *Instance Selection Techniques for Memory-based Collaborative Filtering*. in *SDM*. 2002.

159. Lemire, D. and A. Maclachlan, *Slope one predictors for online rating-based collaborative filtering.* Society for Industrial Mathematics, 2005. **5**: p. 471-480.

160. Webb, B., *Netflix Update: Try This at Home.* 2006.

161. Drineas, P., I. Kerenidis, and P. Raghavan. *Competitive recommendation systems*. in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002. ACM.

162. Barragáns-Martínez, A.B., et al., *A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition.* Information Sciences, 2010. **180**(22): p. 4290-4311.

163. Vozalis, M.G. and K.G. Margaritis, *Using SVD and demographic data for the enhancement of generalized collaborative filtering.* Information Sciences, 2007. **177**(15): p. 3017-3037.

164. Luo, X., Y. Xia, and Q. Zhu, *Incremental collaborative filtering recommender based on regularized matrix factorization.* Knowledge-Based Systems, 2012. **27**: p. 271-280.

165. Chien, Y.-H. and E.I. George. *A bayesian model for collaborative filtering*. in *Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics*. 1999. San Francisco: Morgan Kaufman Publishers,[<http://uncertainty99>. microsoft. com/proceedings. htm].

166. Miyahara, K. and M.J. Pazzani, *Collaborative filtering with the simple Bayesian classifier*, in *PRICAI 2000 Topics in Artificial Intelligence*. 2000, Springer. p. 679-689.

167. Miyahara, K. and M.J. Pazzani, *Improvement of collaborative filtering with the simple Bayesian classifier.* IPSJ Journal, 2002. **43**(11): p. 3429-3437.

168. Park, M.-H., J.-H. Hong, and S.-B. Cho, *Location-based recommendation system using bayesian user’s preference model in mobile devices*, in *Ubiquitous Intelligence and Computing*. 2007, Springer. p. 1130-1139.

169. Xue, G.-R., et al. *Scalable collaborative filtering using cluster-based smoothing*. in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. 2005. ACM.

170. O’Connor, M. and J. Herlocker. *Clustering items for collaborative filtering*. in *Proceedings of the ACM SIGIR workshop on recommender systems*. 1999. UC Berkeley.

171. Chee, S.H.S., J. Han, and K. Wang, *Rectree: An efficient collaborative filtering method*, in *Data Warehousing and Knowledge Discovery*. 2001, Springer. p. 141-151.

172. Hofmann, T. *Probabilistic latent semantic indexing*. in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 1999. ACM.

173. Hofmann, T., *Latent semantic models for collaborative filtering.* ACM Transactions on Information Systems (TOIS), 2004. **22**(1): p. 89-115.

174. Si, L. and R. Jin. *Flexible mixture model for collaborative filtering*. in *ICML*. 2003.

175. Vucetic, S. and Z. Obradovic, *Collaborative filtering using a regression-based approach.* Knowledge and Information Systems, 2005. **7**(1): p. 1-22.

176. Shani, G., R.I. Brafman, and D. Heckerman. *An MDP-based recommender system*. in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. 2002. Morgan Kaufmann Publishers Inc.

177. Honda, K., et al., *Collaborative filtering using principal component analysis and fuzzy clustering*, in *Web Intelligence: Research and Development*. 2001, Springer. p. 394-402.

178. Canny, J. *Collaborative filtering with privacy*. in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. 2002. IEEE.

179. Yager, R.R., *Fuzzy logic methods in recommender systems.* Fuzzy Sets and Systems, 2003. **136**(2): p. 133-149.

180. Al-Shamri, M.Y.H. and K.K. Bharadwaj, *Fuzzy-genetic approach to recommender systems based on a novel hybrid user model.* Expert systems with applications, 2008. **35**(3): p. 1386-1399.

181. Terán, L. and A. Meier, *A Fuzzy Recommender system for eElections*, in *Electronic Government and the Information Systems Perspective*. 2010, Springer. p. 62-76.

182. Porcel, C., A.G. López-Herrera, and E. Herrera-Viedma, *A recommender system for research resources based on fuzzy linguistic modeling.* Expert Systems with Applications, 2009. **36**(3): p. 5173-5183.

183. Ranjbar, M., et al., *An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems.* Engineering Applications of Artificial Intelligence, 2015. **46**: p. 58-66.

184. Seung, D. and L. Lee, *Algorithms for non-negative matrix factorization.* Advances in neural information processing systems, 2001. **13**: p. 556-562.

185. Mnih, A. and R. Salakhutdinov. *Probabilistic matrix factorization*. in *Advances in neural information processing systems*. 2007.

186. Salakhutdinov, R. and A. Mnih. *Bayesian probabilistic matrix factorization using Markov chain Monte Carlo*. in *Proceedings of the 25th international conference on Machine learning*. 2008. ACM.

187. Rendle, S. and L. Schmidt-Thieme. *Online-updating regularized kernel matrix factorization models for large-scale recommender systems*. in *Proceedings of the 2008 ACM conference on Recommender systems*. 2008. ACM.

188. Takács, G., et al. *Investigation of various matrix factorization methods for large recommender systems*. in *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*. 2008. IEEE.

189. Lawrence, N.D. and R. Urtasun. *Non-linear matrix factorization with Gaussian processes*. in *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009. ACM.

190. Koren, Y., R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems.* Computer, 2009. **42**(8): p. 30-37.

191. Luo, X., Y. Xia, and Q. Zhu, *Applying the learning rate adaptation to the matrix factorization based collaborative filtering.* Knowledge-Based Systems, 2012.

192. Berry, M.W., S.T. Dumais, and G.W. O'Brien, *Using linear algebra for intelligent information retrieval.* SIAM review, 1995. **37**(4): p. 573-595.

193. Deerwester, S.C., et al., *Indexing by latent semantic analysis.* JASIS, 1990. **41**(6): p. 391-407.

194. DeCoste, D. *Collaborative prediction using ensembles of maximum margin matrix factorizations*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.

195. Kurucz, M., A.A. Benczúr, and K. Csalogány. *Methods for large scale SVD with missing values*. in *Proceedings of KDD Cup and Workshop*. 2007. Citeseer.

196. Takacs, G., et al. *On the gravity recommendation system*. in *Proceedings of KDD cup and workshop*. 2007.

197. Paterek, A. *Improving regularized singular value decomposition for collaborative filtering*. in *Proceedings of KDD cup and workshop*. 2007.

198. Lee, D.D. and H.S. Seung, *Learning the parts of objects by non-negative matrix factorization.* Nature, 1999. **401**(6755): p. 788-791.

199. Vargas, S. and P. Castells. *Rank and relevance in novelty and diversity metrics for recommender systems*. in *Proceedings of the fifth ACM conference on Recommender systems*. 2011. ACM.

200. Gunawardana, A. and G. Shani, *A survey of accuracy evaluation metrics of recommendation tasks.* The Journal of Machine Learning Research, 2009. **10**: p. 2935-2962.

201. Wen Wu, L.H., Jing Yang. *Evaluating Recommender Systems*. in *ICDIM*. 2012. IEEE.

202. Mobasher, B., et al. *Effective personalization based on association rule discovery from web usage data*. in *Proceedings of the 3rd international workshop on Web information and data management*. 2001. ACM.

203. McNee, S.M., et al. *On the recommending of citations for research papers*. in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*. 2002. ACM.

204. Spearman, C., *The proof and measurement of association between two things.* The American journal of psychology, 1904. **15**(1): p. 72-101.

205. Kendall, M.G., *A new measure of rank correlation.* Biometrika, 1938. **30**(1/2): p. 81-93.

206. Yao, Y., *Measuring retrieval effectiveness based on user preference of documents.* JASIS, 1995. **46**(2): p. 133-145.

207. Cleverdon, C. and M. Kean, *Factors Determining the Performance of Indexing Systems.* URL <http://dspace>. lib. cranfield. ac. uk/handle/1826/863, 1966.

208. Järvelin, K. and J. Kekäläinen, *Cumulated gain-based evaluation of IR techniques.* ACM Transactions on Information Systems (TOIS), 2002. **20**(4): p. 422-446.

209. Moffat, A. and J. Zobel, *Rank-biased precision for measurement of retrieval effectiveness.* ACM Transactions on Information Systems (TOIS), 2008. **27**(1): p. 2.

210. McNee, S.M., J. Riedl, and J.A. Konstan. *Being accurate is not enough: how accuracy metrics have hurt recommender systems*. in *CHI'06 extended abstracts on Human factors in computing systems*. 2006. ACM.

211. Ziegler, C.-N., et al. *Improving recommendation lists through topic diversification*. in *Proceedings of the 14th international conference on World Wide Web*. 2005. ACM.

212. Zhou, T., et al., *Solving the apparent diversity-accuracy dilemma of recommender systems.* Proceedings of the National Academy of Sciences, 2010. **107**(10): p. 4511-4515.

213. Hurley, N. and M. Zhang, *Novelty and diversity in top-N recommendation--analysis and evaluation.* ACM Transactions on Internet Technology (TOIT), 2011. **10**(4): p. 14.

214. Zhang, M. and N. Hurley. *Avoiding monotony: improving the diversity of recommendation lists*. in *Proceedings of the 2008 ACM conference on Recommender systems*. 2008. ACM.

215. Castells, P., S. Vargas, and J. Wang. *Novelty and diversity metrics for recommender systems: choice, discovery and relevance*. in *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*. 2011.

216. Izadi, M., A. Javari, and M. Jalili, *Unifying inconsistent evaluation metrics in recommender systems*, in *ACM International Workshop on Recommender Systems Evaluation: Dimensions and Design*. 2014.

217. Javari, A., M. Izadi, and M. Jalili, *Recommender systems for social networks analysis and mining: precision versus diversity*, in *Complex Systems and Networks*, J. Lü, et al., Editors. 2016, Springer. p. 423-438.

218. Zhou, T., et al., *Accurate and diverse recommendations via eliminating redundant correlations.* New Journal of Physics, 2009. **11**(12): p. 123008.

219. Zhou, T., et al., *Effect of initial configuration on network-based recommendation.* EPL (Europhysics Letters), 2008. **81**(5): p. 58004.

220. Javari, A. and M. Jalili, *Accurate and novel recommendations: an algorithm based on popularity forecasting.* ACM Transactions on Intelligent Systems and Technology, 2014. **5**(4): p. 56.

221. Javari, A. and M. Jalili, *A probabilistic model to resolve diversity-accuracy challenge of recommendation systems.* Knowledge and Information Systems, 2015. **44**: p. 609-627.

1. Netflix is an Internet television network with over 40 million members in different countries. [↑](#footnote-ref-1)
2. http://www.netflixprize.com/ [↑](#footnote-ref-2)
3. www.movielens.com [↑](#footnote-ref-3)
4. www.imdb.com [↑](#footnote-ref-4)
5. http://grouplens.org/datasets/movielens/ [↑](#footnote-ref-5)