Knowledge and Information Systems (2022) 64:35–74
https://doi.org/10.1007/s10115-021-01628-7

SURVEY PAPER

Collaborative filtering recommender systems taxonomy

Harris Papadakis[1] · Antonis Papagrigoriou[1] · Costas Panagiotakis[2,3] · Eleftherios Kosmas[1] · Paraskevi Fragopoulou[1,3]

Received: 5 January 2021 / Revised: 9 November 2021 / Accepted: 12 November 2021 /
Published online: 19 January 2022

Abstract

In the era of internet access, recommender systems try to alleviate the difficulty that consumers face while trying to find items (e.g., services, products, or information) that better match their needs. To do so, a recommender system selects and proposes (possibly unknown) items that may be of interest to some candidate consumer, by predicting her/his preference for this item. Given the diversity of needs between consumers and the enormous variety of items to be recommended, a large set of approaches have been proposed by the research community. This paper provides a review of the approaches proposed in the entire research area of collaborative filtering recommend systems. To facilitate understanding, we provide a categorization of each approach based on the tools and techniques employed, which results to the main contribution of this paper, a collaborative filtering recommender systems taxonomy. This way, the reader acquires a quick and complete understanding of this research area. Finally, we provide a comparison of collaborative filtering recommender systems according to their ability to efficiently handle well-known drawbacks.

Keywords Recommendation systems · Collaborative filtering · Survey · Taxonomy

✉ Harris Papadakis
adanar@hmu.gr

Antonis Papagrigoriou
apapagrigoriou@gmail.com

Costas Panagiotakis
cpanag@hmu.gr

Eleftherios Kosmas
ekosmas@hmu.gr

Paraskevi Fragopoulou
fragopou@ics.forth.gr

[1] Department of Electrical and Computer Engineering, Hellenic Mediterranean University, 71004 Heraklion, Crete, Greece

[2] Department of Management Science and Technology, Hellenic Mediterranean University, 72100 Agios Nikolaos, Crete, Greece

[3] Foundation for Research and Technology-Hellas (FORTH), Institute of Computer Science, 70013 Heraklion, Crete, Greece

1 Introduction

Given the global nature of internet access currently available, product producers and service providers have the unique opportunity to be advertised in an enormous number of customers all over the globe. At the same time, consumers' choices have multiplied exponentially. Even though this allows for a multitude of possibilities and a wider variety of selection, some important implications arise. On one hand, it became increasingly difficult for producers and providers to increase the efficiency of their advertisements, since it became hard to select and target only consumers whose preferences better match their products or services, given the diversity of needs between them. On the other hand, it is not easy for consumers to identify the appropriate products or services that better match their interests, given the diversity and vast availability of options and their time limitations. Both of these result in situations were interesting products/services may pass completely unnoticeable for them. Recommender systems try to amend this situation by analyzing consumer preferences and trying to predict the preference of a user for a single item (e.g., product or service). Recommender systems have a wide range of applications.

The nature of the recommendation problem allows for a diverse set of approaches, and thus, several systems have been proposed to achieve the aforementioned goal. In this work, we present some of the most recent and/or more impactful approaches. Our goal is to provide a review of the systems proposed in the whole research area of collaborative filtering recommender systems and not only on one part of it, so that the reader acquires a quick and complete understanding of this research area. For this reason, our main focus is on presenting the key ideas/techniques used to solve the recommendation problem. More specifically, for each approach, we present an overview of its most representative systems, by selecting some of them and avoiding to present most of their implementation details. Additionally, we encourage and assist the readers that may be interested on finding more details on some of these approaches (or systems), by providing a large number of references. The papers presented in this survey were selected on the basis of their being recent impactful approaches to the collaborative filtering recommender systems problem. With this is mind, we selected articles which were published recently, but have at the same time received a larger number of citations than other papers on the same subject, published in the same year. To illustrate this, we present in Table 1, the average citations of the articles presented in this paper, per year, compared with the average citations of all articles of that year which contained the keywords "collaborative filtering" and "recommender" in their title, obtained through Google Scholar.

1.1 Collaborative filtering-based systems classification

In order to facilitate their understanding, we provide a categorization of each approach based on the tools and techniques employed, which results on the main contribution of this paper that is a recommender systems taxonomy. We remark that several hybrid approaches may

Table 1 Average citations per year, in the field of collaborative filtering recommender systems and the survey's bibliography

Year	2015	2016	2017	2018	2019	2020
General citations	9	21	12	10	4	3
Bibliography citations	110	201	94	89	21	4

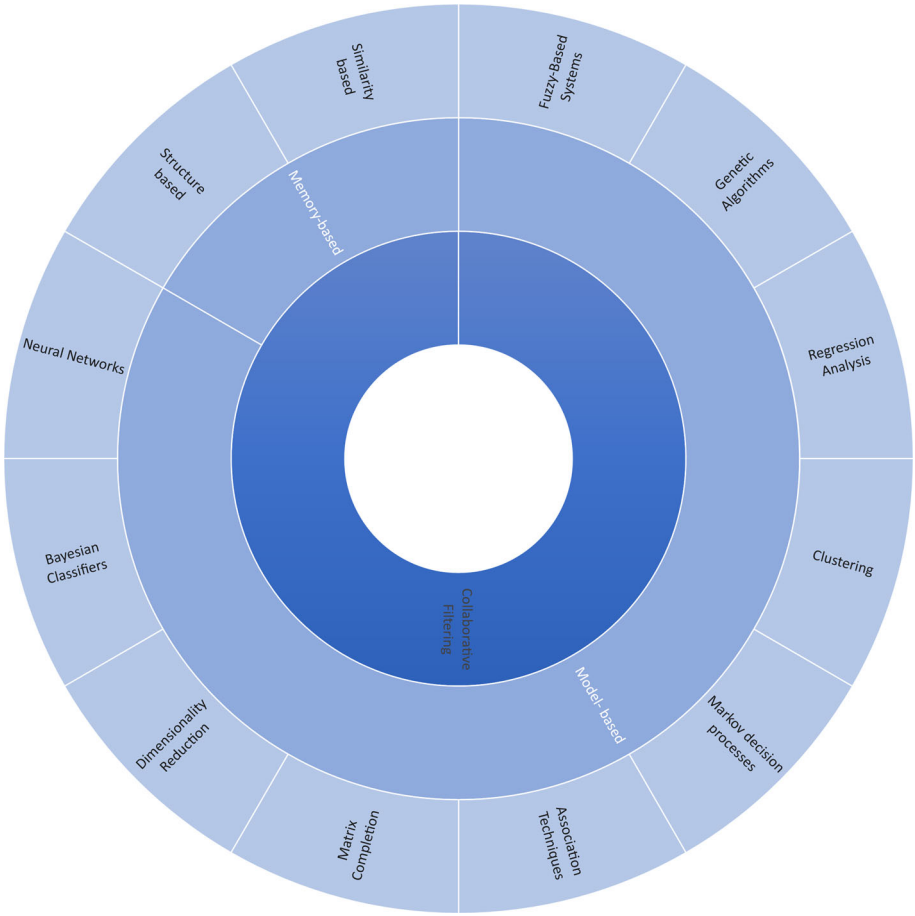


Fig. 1 Hierarchical presentation of content-based recommender system categories

fall into more than one category. Nevertheless, in the context of this document, each of the described approaches has been assigned to a single category, the best fitted one, according to our understanding. Our recommender systems taxonomy is presented in Fig. 1. In the remainder of this section, we briefly describe each of the categories of our taxonomy.

In collaborative filtering, the task is to recommend items to an active user given his preferences and the preferences of other users. In most cases, the preferences are expressed as numerical evaluation scores.

There are two major categories of collaborative filtering methods in recommender systems, namely memory-based and model-based. In the former category, a database of previous users' preferences is maintained and certain calculations are performed on them each time a new recommendation is needed. In the latter category, the same database is maintained again, but a description model is initially developed, using this database, which is then used to make recommendations for an active user [111].

In the following, we briefly describe the subcategories of each category of collaborative filtering, used in our taxonomy. For memory-based collaborative filtering algorithms, we use the following subcategories:

1. *Similarity-based* Similarity-based collaborative filtering algorithms [6,100,113,114] take advantage of the explicit (e.g., ratings) or implicit (e.g., the access frequency to some item's webpage) preferences of the users, in order (i) to discover neighborhoods of similar users or items, by applying similarity functions on their observed preferences, (ii) predict missing preferences (of users on items), by exploiting these neighborhoods and applying prediction functions on the observed preferences of their members, and (iii) use the predicted preferences in order to provide recommendations, usually, by ranking them and proposing the best of them.
2. *Structure-based systems* Structure-based recommender systems [23,97,116] model the problem in the form of a graph and rely on studying the structure of this graph in order to predict the preferences of users for items. In most cases, this graph takes the form of a bipartite user–item graph; however, it can also be user–user or item–item in other scenarios. The structure-based approaches heavily borrow from graph analysis (such as random walks) and link prediction techniques in order to provide the required recommendations.

We now present the subcategories of model-based collaborative filtering algorithms, used by our taxonomy:

1. *Neural networks* A neural network (NN) is a structure of many connected neurons, arranged in layers in systematic ways, which are powerful in processing unstructured data for feature representation. NNs are quite robust with respect to noisy and erroneous datasets of recommender systems and they can alleviate the cold start problem. Several NN architectures have been successfully applied on existing recommender systems.
2. *Bayesian classifiers* Bayesian classifier [40,46] is a probabilistic framework for solving classification problems based on the definition of conditional probability that have been successfully applied on recommender systems. The main advantage of Bayesian classifiers is that they are robust to isolated noise points and irrelevant attributes, and they handle successfully missing values.
3. *Dimensionality reduction* Dimensionality reduction methods [75,76,78,80] reduce the number of input variables by obtaining a set of principal variables that well represent the input data. The resulted reduced representation either compresses the item dimensionality or the user dimensionality into latent factors. This reduced representation can be used to alleviate the sparsity problem for neighborhood-based models.
4. *Matrix completion* Matrix completion techniques [21,24,46,86] recover a matrix from a small sample of its entries and it can be directly applied to predict the unknown values in user–item matrices. The exact matrix completion can be used as a preprocessing step in the correlation method to tackle the sparsity problem.
5. *Association techniques* Association rule mining algorithms [4,46,89] extract rules that predict the occurrence of an item for some user, based on data collected from transactions performed by this user on other items. The key idea in association rule mining is to determine sets of items that are closely correlated in this database of transaction data.
6. *Markov decision processes* Markov decision processes (MDP) [27,95] is a model for sequential stochastic decision problems where outcomes are partly random and partly under the control of a decision maker. In an MDP, there is by definition a four-tuple that consists of the set of states, the set of actions, the reward function that assigns a real value to each state/action pair, and the state-transition function. The problem of predicting a user's behavior has been well solved by using Markov models.
7. *Clustering* Clustering [64,69,71,108,119] methods group similar items or users into separate or overlapped clusters. Clustering techniques have been used either directly or as a preprocessing stage in recommender systems. For example, clusters of similar items

and users can be used in a collaborative filtering approach to determine the most-similar match for a given item or user.

8. *Regression analysis* Regression analysis [68,84,91] is used in a recommender system to fit a line (curve) to the training data points provided by any method, in such a manner that the differences between the distances of data points from the curve or line are minimized. In this way, we can use the distance of this line from any possible data point, in order to obtain a recommendation for this data point.
9. *Genetic algorithms* Genetic algorithms (GA) [8,16,17,88] are search heuristics inspired by Charles Darwin's theory of natural evolution. These algorithms reflect the process of natural selection where the fittest individuals are selected for reproduction in order to produce the offsprings of the next generation. In recommender systems, genetic algorithms have mainly been used in conjunction with other recommendation methods, like clustering and collaborative filtering, in order to improve the accuracy and quality of the recommendations.
10. *Fuzzy-based systems* Fuzzy-based systems [62,103,120,122] employ fuzzy logic and inference techniques in order to provide more accurate recommendations. Most often, these techniques are used in the context of one of the other recommender system approaches described in this document.

In Table 2, we summarize the correspondence of the papers reviewed in this article, to the aforementioned categories.

2 Memory-based collaborative filtering

2.1 Similarity-based systems

Memory-based collaborative filtering (MB-CF) algorithms exploit existing data containing the preferences and, in most cases, the ratings given by users to items, in order to discover similarities between users and/or items. The underlying basic idea of these algorithms is that users with similar reported preferences tend to have similar interests. As a consequence, discovered similarities are used to estimate/predict missing ratings for new items and propose recommendations to users for items they previously ignored.

More specifically, MB-CF algorithms start by constructing a *user–item ratings matrix*, based on the existing ratings. Each row of this matrix contains the ratings given by some user to each of the items she/he has evaluated; so, each row or column of this matrix corresponds to some specific user or item, respectively, and each element r_{uj} of the matrix represents user's u rating for item j. Notice that this matrix contains only user provided ratings for items. For the recommendation process to take place, MB-CF algorithms first try to predict the missing values related to the corresponding *target user* (or *candidate item*), and then to recommend the items (or users) with the largest predicted ratings.

To predict a rating (i.e., a missing matrix value), initially, MB-CF algorithms try to discover relations between users and/or items by applying a *similarity function*. The closely related users to a target user (or items to a candidate item) are also called *neighbors* of the target user (or candidate item); for this reason, MB-CF algorithms are also called *neighborhood-based* collaborative filtering algorithms. Subsequently, the ratings given by (or to) the neighbors are used in order to predict missing matrix values.

According to the type of the predicted relations/similarities, MB-CF algorithms fall into three categories: *user-based*, *item-based*, and *hybrid*:

Table 2 Correspondence of recommender systems articles to categories of the taxonomy

Category	Articles
Similarity-based	[6,10,11,14,22,39,43,48,57,65,83,90,101,110,113,114,126,127]
Structure-based	[23,97,116]
Neural Networks	[12,25,26,28,29,35–37,51,66,82,87,94,105,106,109,115,117,118,123–125]
Bayesian classifiers	[9,40,53]
Dimensionality reduction	[4,13,59,71,71,74–78,80,112]
Matrix completion	[18,49]
Association techniques	[4,89]
Markov decision processes	[27,95]
Clustering	[64,69,71,108,119]
Fuzzy-based	[52,58,62,102,120,122]
Regression analysis	[2,31,68,81,91,121]
Genetic algorithms	[7,8,16,17,56,88]

– A user-based MB-CF algorithm first applies a similarity function between the rows of the ratings matrix, in order to determine the neighbor users of a target user. Then, the prediction of a missing rating for some item by the target user involves the computation of a weighted average (according to the computed similarity between each neighbor user and the target user) of the ratings provided by neighbor users on this item.
– On the other hand, an item-based MB-CF algorithm first applies a similarity function between the columns of the ratings matrix, in order to determine the neighbor items of the candidate item. Subsequently, it uses the ratings given by the candidate user to these neighbor items, in order to predict the preference of the user for the candidate item.
– Finally, a hybrid MB-CF algorithm combines the similarities computed from both user-based and item-based MB-CF algorithms, in order to enhance prediction accuracy.

2.1.1 Advantages, limitations, and considerations

MB-CF algorithms have been extensively studied since they are simple and follow an intuitive approach, which make them easy to implement and debug [3]. Additionally, for the given recommendations, it is often easy to justify why a specific item is recommended.

However, MB-CF algorithms also face limitations. The first such important limitation arises in large-scale settings where the number of users and items is extremely large, which results in considerably high computational cost in terms of time and space complexity, which renders them *impractical*.

Furthermore, user–item ratings matrices contain only the provided user–item ratings. Since the number of items in real-world applications tends to be significantly larger than the number of items rated by users, and since most items are rated by only a small number of users, it follows that user–item ratings matrices maintained by MB-CF algorithms are quite sparse, a problem known as the *data sparsity* problem in recommender systems. For matrices that suffer from this problem, it has been proven [5] that employing conventional similarity measures, such as Pearson correlation coefficient and cosine-based similarity (discussed in Sect. 2.1.2), does not work well, resulting to inadequate recommendations.

Another important limitation of MB-CF algorithms arises when the number of initially available ratings for a particular user or item is relatively small, i.e., during system initialization, or just after a new user or item has just entered the system. This limitation known as the *cold start* problem hinders similarity calculation, since there is not enough information available. New items cannot be recommended until some users have rated them and new users are not provided with accurate recommendations.

Conventional MB-CF algorithms focus only on *accurately* recommending items that are more *relevant* to the target user. However, when recommended items are consistently very similar, a user may not like them, or even worse get annoyed with the repetition of recommendations. On the other hand, when *diversity* is also taken into consideration and items of different types are recommended, then the probability of user satisfaction increases. Recent MB-CF algorithms have targeted the trade-off between recommendation accuracy and diversity. Another consideration that greatly improves users' satisfaction is the *novelty* of recommended items, i.e., items the user has not already seen.

2.1.2 Similarity measures

As explained in Sect. 2.1, an important element of MB-CF algorithms is to determine the neighborhood of a target user or a candidate item, which requires the computation of a similarity measure between users or items, respectively.

Well-known similarity measures include: Pearson correlation coefficient (PCC) [96], constrained PCC [96], cosine-based similarity [11,90], and adjusted cosine-based similarity. In the following paragraphs, we briefly describe each one of them.

User-based: A user-based MB-CF algorithm applies a similarity function between the rows of the ratings matrix, in order to identify users with high similarity to the target user. For two users u and v, the set $I_u \bigcap I_v$ contains the indices of mutually rated items by both u and v.

In user-based MB-CF algorithms, the *Cosine Similarity (COS)* [90] between two users u and v is calculated using the following equation:

$$Sim^{COS}(u,v) = \frac{\sum_{k \in I_u \cap I_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in I_u \cap I_v} r_{uk}^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} r_{vk}^2}} \tag{1}$$

To take into account the fact that different users may have different ratings patterns (some users may be more generous, on average, in their ratings), the *Pearson correlation coefficient (PCC)* [96] measure has been proposed, which mean (or median)-centers ratings in a row-wise fashion:

$$Sim^{PCC}(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \overline{\mu}_u) \cdot (r_{vk} - \overline{\mu}_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \overline{\mu}_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \overline{\mu}_v)^2}} \tag{2}$$

where $\overline{\mu}_u$ and $\overline{\mu}_v$ are the average (or median) ratings given by users u and v, respectively. Because of this bias-adjustment effect, the PCC similarity measure is preferred over the COS measure [3].

It is worth mentioning that the above similarity measures, i.e., COS, ACOS, PCC, and CPCC (median variation of PCC), do not take into consideration the actual number of mutually rated items. As a consequence, they often overestimate the similarities between two users who have mutually rated only a small number of items. One way to solve this problem is by introducing a *similarity weight* [126,127] which reduces the impact resulting from a small number of mutually rated items. The corresponding variants of the above measures are the following:

$$Sim^{A}_{weight}(u,v) = \frac{2 \cdot |I_u \cap I_v|}{|I_u| + |I_v|} \cdot Sim^{A}(u,v) \tag{3}$$

where $A \in \{COS, ACOS, PCC, CPCC\}$. A second solution to the same problem is to reduce the importance of the computed similarity only when the size of the set of mutually rated items is smaller than a particular threshold β. This approach is called *significance weighting* [38,39] and the corresponding variants of the above measures are the following:

$$Sim^{A}_{significance}(u,v) = \frac{min\{|I_u \cap I_v|, \beta\}}{\beta} \cdot Sim^{A}(u,v) \tag{4}$$

where $A \in \{COS, ACOS, PCC, CPCC\}$. A third solution to the same problem is to use a *sigmoid function* [48], which is a continuous monotonically increasing function with output in the range [0, 1]. This function has a smoothing effect on the results and it is used in order to avoid extremely large or small values. Using a sigmoid function, the corresponding variants of the above measures are the following:

$$Sim^{A}_{sigmoid}(u,v) = \frac{1}{1 + \exp\left(-\frac{|I_u \cap I_v|}{2}\right)} \cdot Sim^{A}(u,v) \tag{5}$$

where $A \in \{COS, ACOS, PCC, CPCC\}$.

Item-based: An item-based MB-CF algorithm applies the similarity function between the columns of the ratings matrix, in order to identify items similar to the candidate item. For two items i and j, the set $U_i \cap U_j$ contains the indices of the users who have mutually rated items i and j.

We remark that the similarity measures and their variants presented in Sect. 2.1.2 for user-based MB-CF algorithms can also be used for item-based MB-CF algorithms by simply applying them to columns (items) of the ratings matrix instead of the rows (users). However, in item-based MB-CF algorithms, the ACOS measure provides generally better results [3].

Recently proposed similarity measures: The rating value given by a user to some item reflects its degree of preference. Wu et al.[113] argue that comparing the preferences of two users by calculating the quotient of their corresponding ratings can result to a good similarity measure.

Additionally, we remark that a user rates only part of the available items. In Sect. 2.1.2, we presented a number of variants to the well-known similarity measures that alleviate the impact of a small number of mutually rated items between two users. Tackling the same problem, [114] combined users' similarity, proposed by [113], with a sigmoid function and proposed a new similarity measure called *SigRA*. The user-based SigRA similarity for two users u and v is calculated using the following equation:

$$Sim^{SigRA}(u,v) = \frac{1}{1+\exp\left(-\frac{|I_u|+|I_v|}{2\cdot|I_u\cap I_v|}\right)} \cdot \frac{\sum_{k\in I_u\cap I_v} \frac{\min(r_{uk},r_{vk})}{\max(r_{uk},r_{vk})}}{|I_u \cap I_v|} \tag{6}$$

Notice that all similarity measures proposed up to this point take into consideration only the mutually rated items between two users. However, it has been argued [14] that this does not take into account valuable information, resulting in recommendations that suffer from low diversity and novelty. In order to address this problem, [14] propose an *adjusted similarity* measure that benefits from the unseen items of the corresponding neighborhood of the target user. They make use of a weighting similarity technique, which has the ability to alter the distance between two users, thus affecting the neighborhood formed by a similarity measure. The general formula of this technique is the following:

$$Sim_{weight}(u,v) = W(u,v) \cdot Sim(u,v) \tag{7}$$

where $W(u,v)$ is the corresponding *weight factor* and $Sim(u,v)$ can be any user-based similarity measure, between users u and v. More specifically, they propose a weight factor that represents novelty, considering that the novelty of items outside the items rated by both users may hold valuable information; the latter is what differentiates this measure from previously proposed ones. As a result, they presented the following weight factor:

$$W(u,v) = \frac{Novelty(I_v - (I_v \cap I_u))}{Novelty(I_u - (I_v \cap I_u))} \tag{8}$$

where $Novelty()$ can be any novelty metric, such as the ones presented by [110], namely popularity, expected popularity complement (EPC), expected free discovery (EFD), and expected profile distance (EPD).

Addressing the data sparsity limitation, a *similarity reinforcement* mechanism to enhance MB-CF algorithms is proposed by [43], in order for more potential similarity relationships between users or items to be discovered and more valuable information to be extracted from neighbors with high similarity. To do so, they present a hybrid MB-CF algorithm, which i) integrates user-based similarity reinforcement and item-based similarity reinforcement, and ii) allows for these two to enhance each other. More specifically, the basic idea for user-based

similarity reinforcement is the following (the case of item-based similarity reinforcement is analogous); When two users have no mutually rated items (and thus, conventional similarity measures are not applicable), the sum of the similarities between the items each of these users have rated is used to enhance their similarity; thus, user-based similarity can be enhanced through the affinity of items. This basic idea is formally expressed by the following equation:

$$Sim(u, v) = (1 - \alpha) \cdot Sim(u, v) + \alpha \cdot \frac{\sum_{(i_p, j_q) \in PI_{u,v} \cup NI_{u,v}} W_{pq} \cdot Sim(i_p, j_q)}{\sum_{(i_p, j_q) \in PI_{u,v} \cup NI_{u,v}} |W_{pq}|} \quad (9)$$

where, α, $0 \leq \alpha \leq 1$, is a damping factor controlling the degree that the reinforced user similarity relies on its previous value or the contributions of item similarities, $PI_{u,v} = \{(i_p, j_q) \mid (i_p, j_q) \in I_u \times I_v, sim(i_p, j_q) > 0\}$ includes the item pairs from $I_u \times I_v$ with positive similarities, $NI_{u,v} = \{(i_p, j_q) \mid (i_p, j_q) \in I_u \times I_v, sim(i_p, j_q) < 0\}$ includes the item pairs from $I_u \times I_v$ with negative similarities. W_{pq} weights the contribution of the item pair (i_p, j_q), regarded as the similarity between two ratings and defined as a linear function of the absolute rating difference, as follows:

$$W_{pq} = 1 - 2|r_{ui_p} - r_{vj_p}| \quad (10)$$

Moreover, notice that in the above user-based similarity measure, the similarity of users depends on (and it is enhanced by) the similarity of items; additionally, an item-based similarity measure is defined, in an analogous way, which depends on (and it is enhanced by) the similarity of users. Then, the authors present the *Comprehensive Similarity Reinforcement (CSR)* algorithm, which incorporates these two (dependent on each other) similarity measures in order to produce a matrix U^{re}_{sim} containing the reinforced similarities between the users and a matrix I^{re}_{sim} containing the reinforced similarities between the items. Briefly, this algorithm works as follows: At first, these matrices are initialized using the (user-based and item-based, respectively) PCC similarity measure. Then, iteratively, the above enhanced similarity measures are applied in order to update the elements of U^{re}_{sim} and I^{re}_{sim}, in this order. This iterative procedure terminates either after a predefined number of maximal iterations, or when the variations of both U^{re}_{sim} and I^{re}_{sim} drop below a predefined threshold.

2.1.3 Neighborhood

As explained in Sect. 2.1, during the recommendation process, using the calculated similarities, MB-CF algorithms determine the set of neighbors of a target user or a candidate item, where the final prediction is based on.

However, it has been argued [10,22] that due to data sparsity, this conventional approach leads to high computational complexity and produces bias in the system. To overcome this problem, [10] proposed a new method to calculate the *significant neighbors*, in order to filter the biased nearest neighbors and to improve the accuracy and aggregate diversity for systems incurring data sparsity. We remark that *aggregate diversity* indicates the diversity of items for the entire users set. More specifically, they provide an improved version of a previously proposed similarity measure, called *Relative Similarity Index (RSI)* [101], and apply it to estimate the significant neighbors. We remark that the original RSI metric is a weighted cosine user-based similarity measure (i.e., the RSI similarity value is the result of the multiplication of the COS similarity with some weight), where the weight is calculated from the ratio of the number of mutually rated items over the corresponding users and the maximum number of mutually rated items by any pair of users in the entire dataset. In the improved version of RSI, the denominator of this ratio is the maximum number of mutually

rated items between the target user and any other user; instead of between any pair of users. Finally, the significant neighbors are calculated by ranking the candidate neighbors based on their newly improved RSI similarity value and selecting the top k of them.

We continue by presenting another research work that tries to alleviate the data sparsity limitation following a similar approach with [43], which has been presented in Sect. 2.1.2. Specifically, [57] identifies similarities between users with no mutually rated items, but this time the *subspace clustering* technique is utilized in order to achieve better performance. More specifically, initially, based on the ratings given by users, three new matrices are derived (using the user–item ratings matrix) containing for each user the ratings given to one of the following sets of items, respectively: i) interesting items, i.e., those that have been rated with values 5 or 4, ii) neither interesting nor uninteresting (NIU) items, i.e., those that have been rated with value 3, and iii) uninteresting items, i.e., those that have been rated with values 2 or 1.

Subsequently, multiple item subspaces are constructed that contain mutually rated items by the users and belong to one of the following categories: interesting, NIU, and uninteresting. Each of these item subspaces is constructed using its corresponding derived matrix, based on a known methodology presented by [83], which, however, takes into account only the interesting items. However, [57] argue that NIU and uninteresting rating values are also very important. Subsequently, sets of related users for each item subspace are computed, using any known similarity measure, e.g., PCC or COS.

For each target user u and each category c of items (i.e., $c \in \{interesting, NIU, uninteresting\}$) a tree T_u^c is constructed, as follows: the root of T_u^c is u; the children of u are those users v that participate with u in each item subspace of category c. So u and v are connected since they have mutually rated items in c. Subsequently, v (in the second level of T_u^c) is connected to users in the third level following a similar approach, i.e., v is connected to v' in the third level if they have mutually rated items in category c. This way, users u and v' can be related through T_u^c, although they do not have mutually rated items in c. Using this methodology, the neighborhood of u may be expanded.

Finally, similarity between two users is computed using the constructed trees. Following the notation from the previous paragraph, a conventional similarity measure can be applied between u and v. However, this is not possible for u and v' since they do not have mutually rated items. So, the authors propose a new similarity measure for indirectly connected users, which takes into account the ratings of their common direct neighbor; e.g., the common neighbor v, for u and v'. More specifically, this measure has been derived by combining slightly updated versions of the *Mean Square Difference (MSD)* and the *Jaccard Similarity Coefficient (JSC)* similarity measures [65].

2.1.4 Recommendation

As explained in Sect. 2.1.3, a number of neighbors with improved connections to either the target user or the candidate item are selected. Subsequently, the ratings of these neighbors are exploited in order to provide recommendations.

Prediction functions: One way to use the ratings provided by the selected neighborhood of a user or an item is to estimate a missing rating using a *prediction function*. Following the prediction phase (of some MB-CF algorithm), an ordinary way to recommend items to a target user (or vice versa) is by selecting the top N items with the highest prediction values [3].

In the following paragraphs, we continue by describing the details of some well-known prediction functions. These details differ for user-based and item-based MB-CF algorithms, although similarities may exist.

– *User-based* In user-based MB-CF algorithms, user neighborhoods are constructed. Assume that the rating of user u for item j, i.e., r_{uj}, is missing; so that $\hat{r}_{uj}$ has to be estimated. Let $\mathcal{P}_u^j$ be the set of users in the neighborhood of u who have already provided ratings for item j. For each user $v \in \mathcal{P}_u^j$, rating r_{vj} has been already provided. It is worth mentioning that although during the neighbor selection process for a user u the top-k most related users are chosen, in many cases, this neighborhood contains less than k users; this is mostly common with sparse user–item matrix. In these cases, the cardinality of $\mathcal{P}_u^j$ is less than k.
A first prediction measure is the *weighted average* of ratings, which is calculated using the following equation:

$$\hat{r}_{uj} = \overline{\mu}_u + \frac{\sum_{v \in \mathcal{P}_u^j} Sim(u, v) \cdot (r_{vj} - \overline{\mu}_v)}{\sum_{v \in \mathcal{P}_u^j} |Sim(u, v)|} \tag{11}$$

where the mean rating of u (i.e., $\overline{\mu}_u$) is used to account for differences in user rating patterns, as already mentioned in 2.1.2.

One variant of the above prediction function is the *Z-score*, which further divides the mean-centered value with the corresponding standard deviation. The standard deviation σ_u of the ratings of some user u is calculated using the following equation:

$$\sigma_u = \sqrt{\frac{\sum_{j \in I_u} (r_{uj} - \overline{\mu}_v)^2}{|I_u| - 1}} \tag{12}$$

The equation of the resulting prediction measure is the following:

$$\hat{r}_{uj} = \overline{\mu}_u + \sigma_u \cdot \frac{\sum_{v \in \mathcal{P}_u^j} Sim(u, v) \cdot \frac{(r_{vj} - \overline{\mu}_v)}{\sigma_v}}{\sum_{v \in \mathcal{P}_u^j} |Sim(u, v)|} \tag{13}$$

Notice that the weighted average is multiplied with the standard deviation of u (i.e., σ_u), since (in general) when applying a function during rating normalization, its inverse needs to be applied during the final prediction process.

– *Item-based* In item-based MB-CF algorithms, neighborhoods are constructed for items. Assume that the rating of user u for item j, i.e., r_{uj}, is missing; so, $\hat{r}_{uj}$ has to be estimated. Let $\mathcal{Q}_j^u$ be the items in the neighborhood of candidate item j that has already been rated by user u. So, for each item k, where $k \in \mathcal{Q}_j^u$, it holds that r_{uk} has been already provided. A first prediction measure is the *weighted average* which is calculated using the following equation:

$$\hat{r}_{uj} = \frac{\sum_{k \in \mathcal{Q}_j^u} Sim(j, k) \cdot r_{uk}}{\sum_{k \in \mathcal{Q}_j^u} |Sim(j, k)|} \tag{14}$$

We remark that the prediction measures and their variants presented in Sect. 2.1.4 for user-based MB-CF algorithms can be also adapted for item-based MB-CF algorithms.

Recently proposed: Recently, an alternative to prediction functions has been proposed. Specifically, [6] proposed a method to enhance the accuracy of user-based MB-CF recommendations by replacing the conventional prediction algorithms with the TOPSIS technique, a practical

method for ranking and selecting several externally determined alternatives (i.e., candidate items) based on multiple criteria (i.e., neighbors) through the use of distance metrics [100]. The authors applied TOPSIS in the evaluation and sorting of items rated by nearest-neighbor users in order to produce a set of top-N ranked recommendations.

In more detail, initially, the conventional recommendation process is followed, to determine the neighborhood of the target user. We remark that any similarity measure can be applied in this process. Using the set of items rated by the neighbor users, but not from the target user, a *decision matrix* is constructed with these users in its columns and this set of items in its rows, which contains the corresponding ratings. Subsequently, the TOPSIS technique is applied to the decision matrix in order to evaluate and rank all items included. More specifically, TOPSIS identifies the best alternatives as the one with the shortest and furthest distances form the ideal and negative-ideal solutions, respectively. These two extreme solutions are determined using an updated version of the decision matrix with its values first normalized and then weighted with the calculated similarities between the target user and its neighbors.

2.2 Structure-based systems

Structure-based recommender systems model the problem in the form of a graph and rely on the structure of this graph to predict the preferences of users for items. In most cases, this graph takes the form of a bipartite user–item graph; however, it can also be a user–user or an item–item graph, in other scenarios. The structure-based approaches heavily borrow from graph analysis such as random walks, as well as link prediction techniques to provide the required recommendations. Some of these approaches directly employ the graph structure into the recommendation process, while others first translate the problem into a ranking problem and then use the structure of the graph to rank the nodes. We categorize the approaches presented in this section into three subcategories, namely pure structure-based, ranking-based, and label-based.

2.2.1 Pure structure-based

This family of techniques rely on metrics that analyze the connectivity structure of the graph in order to perform link prediction and thus recommend suitable items. One important aspect to note here is the fact that edges in the bipartite graph are considered to be weightless. A user is connected to an item only if she/he likes the item, otherwise there is no connection. This means that there is no way to exploit degrees of preference. In any case, it is trivial to convert any weighted bipartite graph into a weightless one by using a threshold to eliminate "low preference" edges. The metrics used by pure structure-based algorithms are the following:

1. *Common neighbors:* Given a user u and an item i, this metric measures the common (intersection) items between u and all other users linked to item i. Essentially, an item i is recommended to u only if u and the users that "like" i have many common preferences.
2. *Jaccard coefficient:* This metric is quite similar to the previous one, while it considers ratios of common items, instead of absolute values. This is the ratio of, the number of common items between user u and all other users which "like" item i, over the total number of items "liked" by u and these users.
3. *Adamic/Adar:* The Adamic/Adar metric uses object features to compute the similarity between two objects. The features of some object o, user or item, are defined to be the set of o's neighbors. Consider any user u and any item i; let S be the set of all common

items between u and the other users that "like" i. The Adamic/Adar metric of u and i is defined to be the $\sum_{j \in S} 1/log(deg(j))$, where $deg(j)$ is the number of users that "like" i.

4. *Preferential attachment:* For some user u and some item i, which are nodes of the graph, this metric is defined as $deg(u) * deg(i)$, where $deg(x)$, $x \in \{u, i\}$, is the degree of x, i.e., the number of edges adjacent to x. This metric relates the future interactions with the item's popularity and the users' activity level in the system.
5. *Graph distance:* This metric is defined to be the length of the shortest path between pairs of nodes in the bipartite graph. Thus, an item is recommended to a user if the shortest path distance is small enough (below a threshold).
6. *Katz similarity:* Katz similarity is defined as the sum of weights of all paths between two nodes, exponentially damped by their length. Essentially, an item i is recommended to a user u only if there is a large number of paths of variable lengths leading from u to i.

Metrics 1 to 4 use neighbor sets intersections, while metrics 5 and 6 take into consideration the global graph structure. The aforementioned metrics were first introduced in [63] in the context of social networks and are compared in [44] in the context of recommender systems. The conclusion reached in the paper is that for the first set of metrics, preferential attachment outperformed the rest, while Katz similarity metric outperforms the Graph distance. Finally, the performance of the two best performing metrics was comparable. Li et al. [67] extended the classic structure-based methods, in the sense that content information is added in order to produce a hybrid structure/content-based approach. The classic Jaccard coefficient, for instance, is modified to include a similarity function between users or items. More specifically, the intersection element of the ratio in the Jaccard coefficient is replaced by the sum of all similarity values between the nodes in the intersection, whereas the union is replaced by its cardinality and serves as a "penalty" factor for popular items. Compared to the classic Jaccard coefficient, the new metric is proved to be marginally more accurate in terms of precision and F-measure but clearly more accurate with respect to Recall [67].

Recently proposed metrics: An interesting structure-based approach is described in [116]. In this work, a bipartite graph is used to represent users and items, as well as user preferences in a binary form (like/dislike). However, the graph is modified to include links between nodes of the same type (i.e., users or items) and weights on its links, the values of which are based on the following three formulas:

$$\omega_{similar} = -\omega_{like}^2 \tag{15}$$

$$\omega_{like} = \omega_{similar} * \omega_{like} \tag{16}$$

$$\omega_{similar} = \omega_{similar}^2 \tag{17}$$

Subsequently, the weights on the graph's edges are defined as follows:

$$w(x, y) = \begin{cases} 1, & \text{if x is similar to y} \\ -1, & \text{if x is dissimilar to y} \\ j, & \text{if x likes y or vice versa} \\ -j, & \text{if x dislikes y or vice versa} \end{cases} \tag{18}$$

where x, y are any two adjacent nodes of the graph and $j^2 = -1$.

This adjacency matrix A is used to find shortest paths of various lengths k between nodes, in the form of A^k. In order to make sure those paths connect users to items, only odd values of k are considered. In order to provide recommendations, a top-N item approach per user is

employed on matrix $P(A) = \alpha^1 * A^3 + \alpha^2 * A^5 + \alpha^3 * A^7 + \alpha^4 * A^9 + ...$, with α^p, $p \geq 1$, used as dampening coefficients to reduce the importance of long paths in the final result.

2.2.2 Ranking-based

Ranking-based algorithms rely on transforming the recommendation problem into a ranking problem, while employing the graph structure to rank nodes. These algorithms borrow heavily from classical ranking algorithms such as PageRank [20,72]. The PageRank algorithm calculates and assigns an "importance" weight to each node in a graph, based on the number of times each node is visited during several random walks. Thus, any node's "importance" value is directly correlated to its incoming degree. In addition, in order to avoid dead ends, during each step of the random walk, there is a probability that the walker is "teleported" to a random node. The personalized PageRank algorithm [33,34,107] is modified to include a similarity metric between topics of interests and nodes. The teleportation event is modified to teleport the walker with high probability to a high similarity node of the desired topic, thus increasing the resulting rank of nodes related to that topic. These algorithms are employed in recommender systems in various different ways: By modifying the teleportation step to always teleport the walker to a specific starting node i we can identify and assign "importance" values to the neighbors of i. On social networks, where all nodes are users, this process is used to recommend other users in a straightforward manner. In the case of user–item graphs, there are several ways the personalized PageRank can be employed:

- *Common neighbors:* Given a user u and an item i, this metric measures the common (intersection) items between user u and other users which have item i in common. In essence, an item i is recommended to the user u only if that user and the other users that "like" i have many common preferences.
- A user–user or item–item graph can be also constructed, with weighted edges connecting nodes based on the common rated items (users) or common rated users (items). A personalized PageRank is then employed in the same fashion as before (i.e., in a typical memory-based collaborative filtering manner). In addition, for a single user, one can modify the teleport step to teleport the walker to an item rated by that user, with probability which corresponds to the rated value. The resulting ranking of the items can be used to directly provide recommendations in a top-N fashion.
- Another interesting modification to the PageRank algorithm is called SimRank [50]. PageRank assumes directed graphs (even though it can easily be employed on undirected graphs) and as such whenever it is employed in order to infer similarities between nodes, these similarities are asymmetric. SimRank can be employed whenever symmetric similarities are required and is recursively defined as follows: The SimRank value of nodes i and j is the sum of all the SimRank values of all p, q pairs of nodes (where p any incoming neighbor of i and q any incoming neighbor of j), multiplied by a constant (0, 1) and divided by the product of the cardinalities of the two incoming sets. In the context of recommendations, these similarities can be used in a typical memory-based collaborative filtering approach.

Chiluka et al. [23] employ several previously known structure-based techniques, such as common neighbors, Adamic/Adar and Katz similarity, as well as modifications of the PageRank algorithm to compare their recommendation efficiency. The authors conclude that although neighborhood metrics such as Jaccard coefficient and Adamic/Adar have higher accuracy when a small number of recommendations is required, personalized PageRank has higher accuracy when a large number of recommendations are required per user.

2.2.3 Label-based

Structure-based techniques are also employed as classification mechanisms in label propagation algorithms in order to facilitate recommendations. Such algorithms assume the existence of some metadata information on some nodes (labels) and use structural mechanisms to predict the labels on the rest of the nodes in the graph. This metadata information can then be used in recommending items to the users. Examples of this approach are the following:

- Random walks are used, originating from each node with an unknown label. The label of the first labeled node reached is assigned to the original node.
- Personalized PageRank can also be used, by setting the starting node to be one of a known label. The high ranking nodes of its neighborhood are also assigned the same label.

2.2.4 Hybrid approaches

This hybrid structure/content/model-based approach is presented by [97]. This approach employs a graph of users and items which contains user–item, user–user and item–item edges. User–item edges exist in cases a user "likes" an item and are assigned a weight of 1. User–user edges connect similar users, with a cosine similarity higher than 0.5. Finally, item–item similarity is defined both using cosine similarity of item metadata. Structure-based features (such as common neighbors, Jaccard coefficient, Adamic/Adar, and preferential attachment) are computed, both for randomly connected user–item pairs as well as not connected user–item pairs. These features are then used to train support vector machines, in order to perform link prediction, which in turn leads to the required recommendations.

3 Model-based collaborative filtering

Model-based CF recommendation systems are based on creating a model out of a dataset of ratings. In other words, a model is constructed using information extracted from the dataset. That model in turn is used to make recommendations without having to use the complete dataset every time. This approach has the potential to allow for more efficient solutions in terms of both speed and scalability. Most of the times, model-based CF approaches are either probabilistic (e.g., Bayesian classifiers, clustering) or linear algebra-based (e.g., dimensionality reduction).

3.1 Neural networks

Neural network (NN) is a structure of many connected neurons which are arranged in layers in systematic ways. Neural networks simulate the human brain with the use of neurons. The connections between neurons are determined by weights, that are usually computed during the training phase of the network. Neural networks are quite robust with respect to noisy and erroneous datasets [94]. An open problem, when NNs are used, is to define the optimal network topology for a given application.

NNs have been proven to be capable of approximating any continuous function [42] and more recently deep neural networks (DNNs) have been found to be effective in several domains including computer vision and pattern recognition [60], signal processing and speech recognition [41], compression [32], text processing [28], and recommender systems [12, 25]. The term *deep learning* is used for any NN architecture that optimizes a differentiable

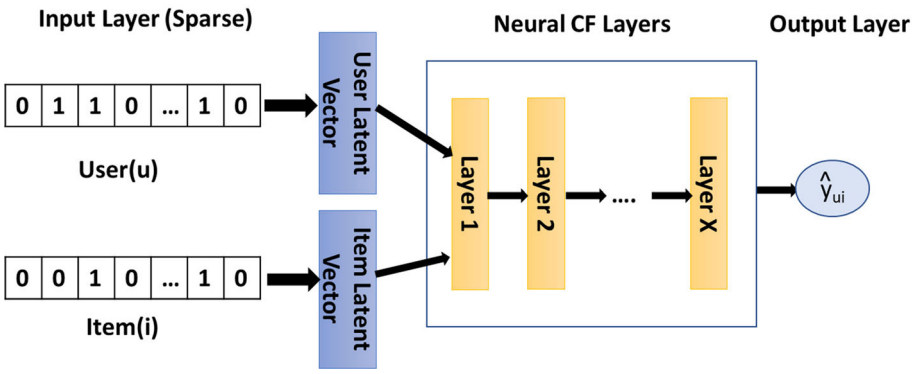


Fig. 2 The architecture of the neural collaborative filtering method [37]

objective function using a variant of stochastic gradient descent (SGD) [125]. In each level, the DNN learns the transformation of its input data into a slightly more abstract and composite representation, learning which features to optimally place in which level, on its own. Although DNNs have exhibited strong ability to learn patterns from dense data [60], the use of DNNs on sparse data is an open problem, since it is not so clear how to employ DNNs for effectively learning feature interactions for sparse datasets. According to [12,25], the future research directions for the deep learning-based RS include their accuracy, scalability, and the use of more datasets. Hereafter, we present several NN architectures that have been successfully used in recommender systems.

– *Multilayer perceptron (MLP)* is a feed-forward neural network with one or more (multiple) hidden layers between the input and output layers, where the perceptron can employ an arbitrary activation function and does not necessarily represent a strictly binary classifier. MLPs can be interpreted as stacked layers of nonlinear transformations, learning hierarchical feature representations [125].

  MLPs show reasonable performance gains over traditional RS methods such as matrix factorization [37]. In order to permit a full neural treatment of collaborative filtering, [37] adopted a multilayer representation to model user–item interactions. Above the input layer, a fully connected layer projects the sparse representation to a dense vector that acts as the latent vector for user (item) in the context of a latent factor model. The user embedding and item embedding are then fed into a multilayer neural architecture, which we term as neural collaborative filtering layers, in order to map the latent vectors to prediction scores. Figure 2 depicts the neural collaborative filtering architecture proposed by [37]. The authors report experimental results on two publicly accessible datasets: MovieLens[1] (movie rating dataset) and Pinterest[2] (content-based image recommendation).
– *Convolutional neural network (CNN)* is a special kind of feedforward neural network with convolution layers and pooling operations that capture the global and local features, performing well in processing data with grid-like topology [60,125]. CNNs are powerful in processing unstructured multimedia data for feature representation learning like audio, text, image, and videos. CNNs are usually applied on the result of a reprocessing step, e.g., outer product [36] or time-frequency representation [109], in order to model the user–item interaction patterns and to capture the high-order correlations among embed-

[1] Movie Lens Datasets available at: http://grouplens.org/datasets/movielens/1m/.

[2] Pinterest Dataset available at: https://sites.google.com/site/xueatalphabeta/academic-projects.

ding dimensions. The correlation matrix obtained by the outer product includes pairwise correlations that facilitate the CNN layers to generalize better and deeper than the fully connected multiple-layer perception [36]. The system proposed by [109] uses as input to the CNN an intermediate time-frequency representation, so it is able to provide recommendations even for new items for which no data is available (cold start problem). The method proposed by [36] has been applied on the two publicly accessible datasets: Yelp[3] (user ratings on businesses) and Gowalla[4] (item recommendations from a location-based social network). The promising experimental results show the successful combination of the outer product to obtain the 2D interaction map and of the CNN in learning high-order correlations among embedding dimensions.

CNNs have been also used to extract features from music signals. [109] compare a traditional approach using a bag-of-words representation of the audio signals with a CNN, and evaluate their predictions on the Million Song Dataset.[5] Generally, CNNs can be combined with feature-based representations (as input) that can alleviate the cold start problem, so they are able to provide recommendations for new and unpopular items [109].

– *Recurrent neural network (RNN)* is suitable for modeling sequential data. Unlike feed-forward neural networks, RNNs have loops and memories in order to remember former computations. Variants such as long short-term memory (LSTM) [82] and gated recurrent unit (GRU) networks [123] are often deployed in practice to overcome the vanishing gradient problem [125].

  The standard LSTM model is not directly applicable to recommendations due to its inability to handle highly sparse data, practical inability to maintain long-range dependencies in user data, and inability to handle irregular time gaps between interactions [82]. The extended LSTM model with a higher-order interaction layer, proposed by [82], is able to handle data sparsity, includes a novel attention mechanism to reduce the burden of encoding the entire user history into a cell vector, and has time aware input and forget gates to handle irregular time gaps between input interactions. The three hidden layer system has been evaluated on Twitter, Google Plus, and YouTube users. The authors used Tweets, Google Plus posts, and YouTube videos with likes or which were added in playlists.

  [115] considered a forward-feedback neural network (FNN) to simulate the collaborative filtering approach and link it with the RNN architecture. In the input layer, one neuron is created for each user. The input to the system is a bitmap indicating whether a user has bought an item or not. For simplicity, all users share the same set of neurons in the hidden layers which also helps the system to identify users' common purchase patterns. The system proposed by [115] has been evaluated on the Kaola testing system using three hidden layers. The first two layers have 1000 neurons and the last layer has only 100 neurons.

– *Restricted Boltzmann machine (RBM)* is a two layer neural network consisting of a visible layer and a hidden layer. It can be easily stacked to a deep net. Restricted here means that there are no intra-layer communications in the visible or the hidden layers [125]. It has been proved that RBM model can infer lower-dimensional representations automatically and is capable to handle large and sparse datasets. Salakhutdinov et al. [87] claim to be the first to propose a recommendation model that built on a neural network based on

[3] Yelp Dataset available at: https://github.com/hexiangnan/sigir16-eals.

[4] Gowalla dataset available at: http://dawenl.github.io/data/gowallapro.zip.

[5] Million song dataset available at: https://labrosa.ee.columbia.edu/millionsong/.

RBM, so that each user has a unique RBM with shared parameters. Since the visible unit of RBM is limited to binary values, binary vectors are used. For example, the vector [0,1,0,0,0] corresponds to a user rating score of two. The proposed RBM system has been trained and tested on the Netflix ratings dataset.

The conditional RBM (CRBM) model is an extension of the basic RBM model with three layers including a visible one, a hidden one, and a conditional layer. The Item Category aware Conditional Restricted Boltzmann Machine Frame model (IC-CRBMF), proposed by [66], employs both RBM and item category information for recommendation. The IC-CRBMF model includes three parts: a visible layer consisting of a rating matrix, a hidden layer containing a hidden feature vector, and a conditional layer consisting of a given category feature vector. Experimental studies on standard benchmark datasets such as MovieLens 100 k and MovieLens 1M have shown its potential on improving recommendation accuracy.

The improved User Occupation aware Conditional Restricted Boltzmann Machine Frame (UO-CRBMF) model, proposed by [117], employs an improved RBM and takes full use of user occupation information by adding a conditional layer with user occupation information.

– *Neural factorization machines (NFM)* combine the linearity of factorization machines (FM) in modeling second-order feature interactions and the nonlinearity of neural networks in modeling higher-order feature interactions. A factorization machine (FM) estimates the target by modeling all interactions between each pair of features. Conceptually, NFM is more expressive than FM, since FM can be seen as a special case of NFM without hidden layers [35]. The key element of NFM's architecture is Bi-Interaction operation, based on which, a neural network model is able to learn more informative feature interactions at the lower level. In [35], a neural network model NFM has been used for sparse predictive analytics. Neural hierarchical factorization machine [118] inherits the advantages of NFM on modeling second-order and higher-order feature interactions. Additionally, it also could model the hierarchical structure information yielding high performance results on user's event sequence analysis.
– *Deep autoencoders* is a feedforward network, which tends to learn a distributed representation of data that is very powerful in dealing with noisy data to learn the complex and hierarchical structure from the input data. It is specifically used for learning dimensionality reduction of the set [25]. It essentially possess one hidden layer between the input and output layers with the more compact representation of hidden layers than input and output layers. Autoencoder variants such as sparsed-autoencoder [105] and stacked-autoencoder [106] have been proposed to provide robust feature representation for personalized recommendations. [25] show that autoencoder models are the most widely exploited deep learning architectures for RS followed by the CNN and the RNN models. [51] presents a fast autoencoder-based model for analyzing high-dimensional and sparse matrices data for RS. It focuses on the known data to reduce the computational cost and avoid hypothetical bias.

  Recently, an MF-based approach for trust-aware recommendation in social networks has been proposed [26]. More specifically, since the recommendations of MF are affected by the initialization of the user and item latent vectors, authors utilize deep autoencoder to pretrain the initial values of their learning model. Additionally, their learning model incorporates recommendations from trusted friends and also involves community's effect.
– *Deep hybrid models for recommendation* integrates neural building blocks to formalize more powerful and expressive models. For example, [124] and [29] proposed a CNN and RNN based hybrid model for hashtag and citation recommendations, respectively. [117]

combined the proposed UO-CRBMF model with the IC-CRBMF model [66] to improve recommendation accuracy. The hybrid CRBMF model is implemented as a weighted linear combination on the scores of the UO-CRBMF item-based and the UO-CRBMF user-based model.

3.2 Bayesian classifiers

A Bayesian classifier is a probabilistic framework for solving classification problems and it has also been applied on recommender systems [46]. It is based on the definition of conditional probability and the Bayes theorem [53] that describes the probability of an event, based on prior knowledge of conditions related to the event. So, the (posterior) probability of a model, given the data, is proportional to the product of the likelihood times the prior probability. The likelihood component includes the effect of the data while the prior specifies the belief in the model before the data was observed [9].

The main advantages of Bayesian classifiers are that they are robust to isolated noise points and irrelevant attributes, and they handle missing values by ignoring the instance during probability estimate calculations [9].

A recommender system based on a completely Bayesian probabilistic framework that factorizes the rating matrix into two nonnegative matrices has been proposed by [40]. The proposed method associates a vector to each user and each item. The components of these vectors lie within [0, 1] with an understandable probabilistic meaning, that allows to identify groups of users sharing the same tastes, and justify the recommendations in the same way as the K-NN algorithm. The method has been evaluated on publicity available datasets like MovieLens 1M, MovieLens 10M, and NetFlix.

3.3 Dimensionality reduction

Dimensionality reduction methods reduce the number of input variables by obtaining a set of principal variables that well represent the input data. The dimensionality reduction can be achieved:

- by keeping only the most relevant variables from the original dataset (this technique is called feature selection) or
- by finding a smaller set of new variables that contain basically the same information as the input variables [104].

The reduced representation, that is provided by the dimensionality reduction methods, will either compress the items' dimensionality or the users' dimensionality into latent factors. This reduced representation can be used to alleviate the sparsity problem for neighborhood-based models. Additionally, the similarity computations in the reduced representation are more robust because the new low-dimensional vector is fully specified [4].

Dimensionality reduction techniques are a type of the matrix factorization method. Matrix factorization approaches have been quite popular in the field of recommender systems. One such algorithm was the big winner of the Netflix Prize competition in 2006 [13]. Matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. So, matrix factorization map both users and items to a new joint latent factor space [59]. Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users [59]. For example, the discovered factors of users could be the dimensions female- versus male-oriented and serious versus escapist.

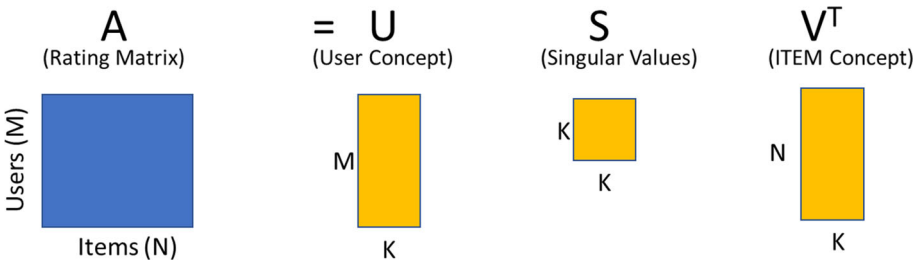


Fig. 3 The SVD schema applied on the user–item recommender problem [71]

For this latent factor model, a user's predicted rating for a movie would equal the dot product of the movie's and user's locations on the graph.

SCoR [80] is a synthetic coordinate-based recommendation system that assigns synthetic coordinates (vectors) to nodes (users and items) as proposed by matrix factorization. Instead of using the dot product, SCoR uses the Euclidean distance between a user and an item vectors to provide an accurate prediction of the user's preference for that item. The proposed framework has several benefits. It is parameter free, thus requiring no fine tuning to achieve high performance, and is more resistance to the cold-start problem. In [76], a stage called UTEC is introduced to improve the SCoR recommendations, taking into account the error on the training set between users and items and their proximity in the synthetic Euclidean space of SCoR. The synthetic coordinates have been also successfully applied to the distributed community detection problem [79], personalized video summarization [77], detection of abnormal profiles in RS [75,78], and to the interactive image segmentation problem [74] providing high performance results compared to other state-of-the-art methods on public datasets. SCoR outperforms nine other state-of-the-art recommender systems against four real datasets, including a brief version of the dataset used in the Netflix challenge [13].

Singular value decomposition (SVD) is a factorization of a matrix into the product of three matrices U, D, and V, where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries [71]. The SVD can be used to find the most similar items and users in each cluster of items and users. This can significantly improve the scalability of the recommendation method. Nilashi et al. [71] used the expectation maximization clustering and non-incremental SVD for dimensionality reduction tasks. Figure 3 shows the SVD schema in user–item matrix in CF where A denotes the rating matrix of user to items, U refers to user concepts matrix, S denotes singular values, and V^T denotes a reprehensive of item concepts. The method proposed by [71] has been evaluated under the MovieLens and Yahoo! Webscope R4 datasets.[6] In both datasets, the users have provided ratings on a 5-star scale (1 to 5).

Finally, a recent hybrid work is described in [112], which uses dimensionality reduction in multiple facets of the problem, by applying it on a heterogeneous information network. Graph embedding is used both in the structural information of the HIN, as well as the content metadata information of each item. In addidion, HIN incorporates session information (i.e., which items were accessed by the user at the same time), therefore leading to a completely hybrid collaborative filtering system which also incorporates content and contextual information. Its hybrid approach also makes the proposed mechanism resilient to the cold-start problem.

[6] Yahoo! Webscope R4 dataset available at: http://webscope.sandbox.yahoo.com.

3.4 Matrix completion

The goal of a matrix completion technique is to recover a matrix from a small sample of its entries, and comes up in many areas of science and engineering including machine learning, control, remote sensing, and computer vision [21], as well as in RS. The technique can be directly applied to predict the unknown values within the user–item matrices [21,46]. An unknown $n \times n$ matrix of low rank r can be recovered from just about $nrlog^2n$ noisy samples with an error which is proportional to the noise level [21]. A popular approach for matrix completion is the alternating least square (ALS) method [24,86]. According to this method, the recommendation matrix R is modeled by the product of two low rank matrices U and V^T that are estimated in an iterative schema (e.g., gradient descent based) in order to minimize the least squares error of the observed ratings.

The exact matrix completion can be used as a preprocessing step in the correlation method to tackle the sparseness problem [18]. This can be done via a correlation method that selects a subset of users based on their similarity to the active user, and consequently, the weighted combination of their ratings is used to provide the prediction of rating for the active user. Since the lower number of the missing values in the data matrix leads to a higher level of accuracy in choosing more similar neighbors for the active user, it holds that the matrix completion stage improves the accuracy of the correlation method. The method proposed by [18] has been evaluated on the publicity available EachMovie dataset.

There exist several quality factors of recommender systems which are not covered by the matrix completion task. According to [49], matrix completion algorithms that predict well on held-out ratings that users provided, may predict poorly on a random set of items that the user has not rated. This can mean that algorithms which are tuned to perform well on past ratings are not the best algorithms for recommending in the real world.

3.5 Association techniques

Association rule mining algorithms extract rules that predict the occurrence of an item for some user based on data collected from transactions performed by this user on other items. The key idea in association rule mining is to determine sets of items that are closely correlated in this database of transaction data. Association rules can form a very compact representation of preference data that may improve efficiency of storage as well as prediction accuracy [4,46]. Association rules can also be used on unary ratings matrices, where a user specify a liking for an item, but there does not exist any mechanism to specify a dislike. For example, the items bought by a user can be set to 1, while the rest of the items are approximately set to 0 [4].

An association rule is said to be fired by a user, if the set of items in the antecedent of the rule is a subset of the items preferred by that user. According to this methodology, all the *fired* association rules of a given user have to be determined. All of the *fired* rules are then sorted in order of reducing confidence. The first k items of these sorted rules are used to recommend the top-k items to the given user [4,89].

3.6 Markov decision processes

Markov decision processes (MDP) is a model for sequential stochastic decision problems where outcomes are partly random and partly under the control of a decision maker. MDP are used in applications where an autonomous agent is influencing its surrounding environment

through actions (e.g., navigating robot) [95]. Usually, an MDP is by definition a four-tuple that consists of the set of states, the set of actions, the reward function that assigns a real value to each state/action pair, and the state-transition function, which provides the probability of a transition between every pair of states given each action. An optimal solution to the MDP is such a maximizing reward stream. Thus, the Markov model relies on the short memory assumption of typical user action sequences. The order of a Markov model shows how to look far into the past and is related to the model complexity.

The problem of predicting a user's behavior on a website has been studied using Markov models [27,95]. In order to provide the kind of performance required by an online commercial site, various approximations and, in particular, heavy use of the special properties of the proposed state space and its sequential origin should be done. Whereas the applicability of these techniques beyond recommender systems is not clear, this application represents an interesting case study of a successful real system.

A disadvantage of Markov models is that as the order of models becomes higher, then the performance, as well as the space and runtime requirements, increase. However, it is possible to select parts of different order Markov models so that the resulting model has a reduced state complexity [27] via pruning the states without affecting the performance of the overall scheme. According to the experimental results presented by [27], a prune up to 90% of the states from the initial high-order Markov model improves the accuracy by up to 11%.

3.7 Clustering

Clustering methods group similar items or users into separate or overlapped clusters. They have been used in many applications of pattern recognition, computer vision, image processing, and data mining [47,73]. Clustering techniques have been used either directly [64,108] or as a preprocessing stage in recommender systems [71,119]. For example, clusters of similar items and users can be used in a CF approach to determine the most-similar match for a given item or user. The method proposed by [64] tested a clustering method for item-to-item collaborative filtering system employed at Amazon.com by building a similarity-items table. The method proposed by [119] uses k-means clustering to smooth the unrated data of individual users, thus solving missing-value problems. The method has been experimentally evaluated on MovieLens and EachMovie datasets.

Tsai and Hung [108] used two well-known clustering techniques, namely self-organizing maps (SOM) and k-means, for three cluster ensemble methodsl, namely cluster-based similarity partitioning algorithm, hypergraph partitioning algorithm, and majority voting, which are used to combine multiple k-means and SOM models. The experimental results based on the Movielens dataset show that cluster ensembles can provide better recommendation performance than single clustering techniques in terms of recommendation accuracy and precision.

A model-based collaborative filtering method [69] is applying a novel graph clustering algorithm and it is also considering trust statements. So, the given user/item set is mapped into a weighted graph, where the set of users/items define the graph nodes, and the similarity value between each pair of the users/items define the weights between the corresponding edges. Then, a sparsest subgraph finding algorithm is applied on the graph to initialize the cluster centers for the clustering method. The proposed graph clustering method consists of the following three steps.

1. Finding initial cluster centers,
2. Modifying cluster centers and merging clusters, and

3. For each unseen item, a rate is predicted based on the clusters and the top-N interested items are recommended to the active user.

3.8 Fuzzy-based systems

The main goal of recommender systems is the prediction of a user's preference on an unrated item. In order to achieve this, a diverse set of techniques and approaches is used in the literature, as this document clearly demonstrates. Similarly, these approaches exploit a diverse set of available information, which is very often not defined in a precise way. In addition, since users are humans, their preferences are again not well defined and always consistent. For these reasons, many approaches employ fuzzy interference techniques to compensate these factors and provide more accurate predictions. These fuzzy techniques are usually employed within the context of a more traditional approach. The remainder of this section presents such recent and/or illustrative approaches.

3.8.1 Memory-based

A classic use of fuzzy logic in RS is comprised of expressing the known user preferences (ratings) in a fuzzy fashion. The approach presented by [120] is basically memory-based CF system. However, the ratings are expressed in fuzzy set memberships, namely strongly like (SL), like (L), indifferent (I), dislike (D), and strongly dislike (SD), and similarity function is based on metadata (genre-based) memberships (i.e., membership degree for each type of genre). In addition, minimal content information for items is used in the form of the genre attribute, with various values for each fuzzy set. Each item has a membership function for each set (genre type). The fuzzy memberships are used to calculate similarities between items. Ratings, which are expressed as fuzzy set memberships, take into account the average rating values of a user. Similarly with conventional memory-based CF algorithms, recommendation for user u and item i is the weighted average of (fuzzy) rating for each item rated by u, multiplied by the (fuzzy) similarity of that item to i.

The next approach [122] is based on the same principles as the previous one, with a difference in the top level process, used to generate recommendations. Again, numeric ratings of users to items are fuzzified to fuzzy set memberships (i.e., SL, L, I, D, and SD), while a fuzzified Pearson-based similarity metric is defined, which calculates item–item and user–user similarities using these fuzzified ratings. More specifically, recommendations are generated as follows. Initially, ratings are fuzzified and the item–item similarity function is used, to select top-K neighbors for each item. Then, item–item collaborative filtering is used to predict fuzzy ratings for each item, in order to make the dataset denser. Finally, the user–user similarity function is used to select top-K neighbors for each user and user–user collaborative filtering is used to predict fuzzy ratings , i.e., the recommendations for each user.

Similarly, [62] initially fuzzify the ratings as in "preference fuzzy sets," namely like (L), neutral (N), and dislike (D), and then, the support of an itemset (i.e., number of transactions that support a rule) is calculated based on the item-fuzzy set pairs. Considering an example, for any item $I1$, let the pair $\langle I1, D\rangle$ be the value of an itemset. Then, without fuzziness, this value indicates the ratio of users that disliked $I1$. While, when introducing fuzziness, this value is calculated using the degree of membership of the preference of each user to $I1$ to the fuzzy set of dislikes (D); this is called "Fuzzy Support" (FS). The FS of $\langle\{I1, I2\}, \{D, L\}\rangle$ is calculated by multiplying the corresponding membership values of $\langle I1, D\rangle$ and $\langle I2, L\rangle$,

where $I2$ in another item, for each user and adding the products, while finally dividing by the number of users. Association rules are then created in the form $\langle A, X\rangle \rightarrow \langle B, Y\rangle$, where A, B are items and X, Y are preference fuzzy sets (e.g., $\langle I1, D\rangle$->$\langle I2, L\rangle$). Confidence of the rules is calculated in normal fashion, using the modified support formula: $FS(\langle\{I1, I2\}, \{D, L\}\rangle)/FS(\langle I1, D\rangle)$. Fuzzy variances of correlation and variance statistical metrics are also defined. These variances are also based on the FS formula, described above.

Son [103] use user metadata in the form of demographic information, such as age and standard of living. This information is fuzzified, and then, it is used to calculate user–user fuzzy similarity, based on these demographic data. Additionally, traditional Pearson-based user–user similarity is calculated using user ratings. Then, these two similarities are combined in a weighted fashion, in order to provide the final similarity, which is used in a conventional memory-based (user–user) CF fashion.

3.8.2 Content-based

A typical example of employing fuzzy logic in recommender systems can be found in [102]. Fuzzification is used in two of the system's aspects, namely in the item attributes (metadata) as well as contextual information. With regard to the first one, the authors use a well-accepted typology of tourist roles (extreme sports, fun and leisure and so forth). Items (points-of-interest) are described based on their relevance to the aforementioned roles defined in the typology, in a fuzzy fashion. Moreover, each user creates a profile by explicitly providing his/her preference to each role (essentially, his/her degree of membership to each role). In addition, contextual information is used and similarly fuzzified (for instance, weather state or available time). Manually provided fuzzy rules are used to match user profiles to items.

A clustering-based approach is presented by [58]. The authors firstly demonstrate that clustering techniques can be used to alleviate the shortcomings of Memory-based CF algorithms' similarity functions in case of sparse dataset, by factoring in the similarity function, user (or item) membership. In addition, they demonstrate that the fuzzy C-means clustering algorithm outperforms classical clustering algorithms in this context, mainly due to the fact that it allows for varying degrees of memberships in multiple clusters, per user.

3.8.3 Model-based

Kant and Bharadwaj [52] try to rectify the main weak point of memory-based CF systems, with a similarity function that captures the intricacies of the specific case domain successfully, by employing naïve Bayesian classifier (NBC). This NBC is further enhanced by using its fuzzy equivalent, a classifier that allows for fuzzy values of attributes as well as classes. This fuzzy Bayesian classifier is used by two of the proposed approaches as well as by a third combinatory one. The first approach fuzzifies the user ratings into two fuzzy sets (liked/disliked). The FBC is then used to classify (liked/disliked) a new item for a user, using the fuzzified user ratings as attributes. The second approach requires the existence of item description features. These features are fuzzified to illustrate the presence of each and every available feature for each item. The FBC is then used to classify (liked/disliked) a new item for a user, using the fuzzified item features as attributes. The second approach is obviously content-based, which means it is less susceptible to the cold-start problem. Finally, a third hybrid approach is presented, which merges the previous two approaches, by generating independent recommendations from both these approaches and merging them

with a weighting factor, which is dynamically adjusted to assign a higher weight to the most successful approach, on a per case basis, each time a recommendation is requested.

3.9 Regression analysis

Regression analysis is a form of statistical modeling technique which investigates the relationship between dependent and independent variables. This technique is used for many other uses other than recommendation systems, like forecasting, time series modeling and finding the causal effect relationship between the variables. Regression analysis is an important tool for modeling and analyzing data; it helps one to understand how the typical value of the dependent variable/s (or 'criterion variable') changes when one or more of the independent variables is varied. The concept of regression analysis is to fit a curve (or a line) to the data points, in such a manner that the differences between the distances of data points from the curve or line are minimized. In this way, the distance of this curve/line from a set of possible data points (recommendations) can be used, in order to obtain a recommendation set for the user [84].

Several regression analysis techniques exist, which make predictions. In most cases, any (but only one) of the independent variables can vary. One of the most studied regression analysis techniques is linear regression. This is because models which depend linearly on their unknown parameters are easier to fit, than models which are nonlinearly related to their parameters. The reason is that the statistical properties of the resulting estimators are easier to determine. In most recommendation systems that use collaboration filtering, patterns exist among different users' preferences. These patterns can be used to predict more accurate recommendations using a regression model, compared to other techniques like weighted average [31]. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response, given the values of the predictors, rather than on the joint probability distribution of all of these variables. That is the domain of multivariate analysis [85].

A common example of recommendation using linear regression analysis is to predict a movie's rating for a specific user (dependent variable) based on her/his weighted ratings on other movies (independent variables). The models differ in the way the weights are calculated. However, all models use the ratings of other customers for weight estimation. After estimation of the model parameters, recommendations from the models can be received by transforming the predictions into discrete ratings on a 6 point scale [68,91]. Another example of model regression is to use Bayesian hierarchical modeling in conjunction with a content-based recommendation system. In content-based recommendation systems, the profile learned for a particular user is usually of low quality when the amount of data from that particular user is small. This is known as the "cold start" problem. This means that any new user must endure poor initial performance until sufficient feedback from that user is provided. The improvement of the recommendation system performance for a particular user in cold start cases is done by exploiting information from other users through a Bayesian hierarchical modeling approach [121].

Other regression models with only one independent variable to vary are:

- nonlinear regression models, where the model function is not linear in the parameters and the sum of squares must be minimized by an iterative procedure,
- interpolation, and extrapolation models, where the models predict a value of some variable Y given known values of other variables. In case, the prediction is within the range of

values in the data-set, then these models are known informally as interpolation models; otherwise, they are known as extrapolation models, and
– Bayesian linear regression models, which are similar to linear regression models, while the statistical analysis is undertaken within the context of Bayesian inference.

There are also cases of multivariate regression analysis in which more than one independent variable can vary. In this case, the dependent and independent variables are expressed as arrays, in the model function. Additionally, logistic regression and ridge regression are used in recommendation systems to predict an items's rating for a specific user (dependent variable) based on her/his weighted ratings on other items (independent variables). These models differ in the way the corresponding weights are calculated [68].

However, as already mentioned, not only content-based recommendation systems, but also collaborative filtering-based recommendation systems suffer from the "cold start" problem, when the available rating data are extremely sparse. To handle this (cold-start) problem, the user and item features are exploited, additionally to the user ratings. This is done by constructing tensor profiles for user–item pairs from their individual features. Within the tensor regression framework, optimization of the regression coefficients is done, by minimizing pairwise preference loss. In this way, reasonable recommendation can be provided even for new users with no historical ratings but only a few demographic information [81].

Additionally to the cold-start, there is also the warm-start problem, where too much data exists during the initiation phase of a recommendation process, resulting in less accurate recommendation. As a solution to the warm start problem, regression-based latent factor model (RLFM) techniques [2] can be used. These techniques work as follows:

– Initially, they improve prediction for old user–item tuples by simultaneously incorporating features and past interactions and then,
– provide predictions for new tuples through features.

3.10 Genetic algorithms

Genetic algorithms are search heuristics inspired by Charles Darwin's theory of natural evolution. These algorithms reflect the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. These algorithms are intelligent exploitation of random search. They are provided with historical data, so that the search is directed into the region of better performance, in solution space. In recommender systems, the genetic algorithms are utilized in order to generate high-quality solutions and prediction optimizations. A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain.

Genetic algorithm-based recommender systems more commonly use a representation of an array of bits for each of their solution candidate. Arrays of other types and structures can also be used, similarly. The main property of this representation is that the array parts are easily aligned due to their fixed size, and this makes these representations convenient to be used with other recommendation algorithms in hybrid RS. Other solutions may also be used, like variable length representations, tree-like representations, and graph-form representations. Once the genetic representation and the fitness function are defined, a genetic algorithm proceeds to initialize a population of solutions and improve them. In a model-based CF recommendation system, the historical users/items data is used to create a model for a genetic algorithm in order to generate the recommendation [17].

In recommender systems, genetic algorithms have mainly been used in clustering [56], hybrid user models [8], and genetic algorithms-based CF-based recommender systems which

does not require any additional information [16]. Generally, the genetic algorithms are used in order to improve the quality of recommendation, generating a more accurate and better quality recommendation [7]. Genetic algorithms can be used in conjunction with other algorithms, to design more efficient RSs. For example, [88] developed a hybrid RS for learning materials. Their RS consists of attribute-based filtering and genetic-based RS in order to improve the quality of recommendation in the e-learning environment.

4 Evaluating collaborative filtering recommender systems

In this section, initially we provide an overview of the strengths and weaknesses of the collaborative filtering-based recommender systems. Afterward, we present a comparison for the various categories in our taxonomy based upon the strengths and weaknesses references.

4.1 Overview of strengths and weaknesses of the collaborative filtering recommender systems

In this subsection, we provide an overview of the strengths and weaknesses of the collaborative filtering recommender systems based on the studied bibliography.

One of the greatest advantages of collaborative filtering recommender systems is that they rely on the interactions between users and the set of items they interact with. This provides to these recommender systems a number of advantages [30].

- *Not required to understand item content* The content of the items does not necessarily tell the whole story, such as movie type/genre, and so on. By using only information based on the interactions between the users and the set of items, there is no need for better understanding the item content.
- *Captures inherent subtle/latent characteristics* This is very true for latent factor models. If most users buy two visible unrelated products a latent association between these can be extracted.

But from the way these methods operate, there are some disadvantages.

- *Cold start* The cold start problem occurs, when a new user or item just enters the system. There are three kinds of cold start problems: the new user problem, the new item problem, and the new system problem.
- *Sparsity* The sparsity problem is one of the major problems encountered by CF recommended systems, where data sparsity has great influence on the quality of the provided recommendations. The main reason behind data sparsity is that most users only rate a small subset of the items, thus the available ratings are usually sparse. Collaborative filtering methods suffer from this problem since they usually depend on the ratings matrix to make proper recommendations [46,98].
- *Over specialization/filter-bubble* In collaborative filtering, the over specialization problem prevents users from discovering new items and other available options. This means that users are restricted to recommendations which resemble those already available to them or directly related to their profiles [70].
- *Scalability* Scalability measures the ability of a system to work effectively with high performance even when the availability of information grows significantly. But with the enormous growth in information over the years, recommender systems have to cope

with an explosion of data. In CF recommender systems, computation overhead grows exponentially, sometimes leading to inaccurate results [46].
– *Synonymy* The problem of synonymy arises when an item is represented with two or more names or entries with similar meanings. In these cases, the CF recommender system cannot identify whether the terms represent different items or the same item; therefore, the recommendation does not consider the latent association between them. This is the reason why new items are not recommended as soon as they are rated by users.

4.2 Comparison of the contented based recommender systems categories

In this subsection, we provide an overview/comparison of the strengths and weaknesses of the various categories in our taxonomy. The comparison is based on the studied bibliography, with regard to the most important open issues in the field today.

The comparison illustrated in Table 3 is based on the work performed by [1,15,45,46,54, 61,70,92,93,99], and [3].

The categories that define the columns of Table 3 comprise the most popularly identified open issues in Recommender Systems literature [55]. It should be noted here that we did not perform any evaluating experiments ourselves, but rather, consistent with the philosophy of a survey paper, we summarize here the aggregate information reported in the various articles of the literature mentioned above. The evaluation scores (good, average, and poor) are by no means quantitative, since we could not identify any articles that compare those various issues in an experimental fashion, but rather represent the reported consensus in the studied bibliography.

From the contents of the table, we draw the conclusion that the cold start problem still is an important issue for collaborating filtering recommender systems. Especially in memory-based CF-RS [92], in case of a new user, there is very little information available about the user, similarly for a new item, no ratings are usually available, which makes it difficult to extract useful recommendations [46]. Exception to the above constitute CF methods that include regression analysis, where additionally to the ratings, user and item features are exploited, and as a result there is improved behavior in the cold start problem [81,121].

Sparsity also remains an important issue. Some CF methods exhibit improved behavior with respect to sparsity. These include methods which use multidimensional recommendation models like Bayesian classifiers [54] or dimensionality reduction techniques [1] like SVD [15]. Data sparsity is also addressed by integrated multi-task learning, integrated CNN and RNN and other neural network techniques [25].

Just like sparsity, filter bubble also remains an important issue, despite recent efforts in reducing their effects. After coping with the diversity problem using genetic algorithms, users can be provided with a set of diverse recommendations and a wide range of alternatives [98].

Scalability, as expected, appears to be heavily dependent on the particularities of each approach. To cope with this problem, in CF recommender systems different techniques have been employed [54,93], like clustering, reducing dimensionality [15], and Bayesian networks.

Finally, synonymy is an issue present in most approaches, but in some CF methods, which take into consideration the probability of a word w_i to be conditioned only on the current tag, the problem of synonymy can alleviate. These include dimensionality reduction with latent factors [15] and more specifically methods using single value decomposition (SVD) [54].

Figure 4 depicts the percentage of RS sub-categories that perform good, average, poor, and unknown performance over the five open issues studied in this research based on the results of Table 3. The figure clearly identifies the diversity (filter bubble) problem as the most

Table 3 An overview of collaborative filtering RS categories sensitivity, with respect to five different known RS drawbacks

RS category	Cold start	Sparsity	Filter bubble	Scalability	Synonymy
Similarity-based	–	–	–	–	–
Structure-based systems	–	+ [4]	–	v [4]	–
Neural networks	+ [15]	+ [1]	–	–	–
Bayesian classifiers	–	v [46]	–	+ [46,54]	–
Dimensionality reduction	–	v [1,4,15]	–	+ [15,46,54]	+ [46,54]
Matrix completion	–	–	–	–	–
Association techniques	–	–	–	–	–
Markov decision processes	–	–	–	–	–
Clustering	–	–	–	v [1,4,54]	–
Regression analysis	v [15,61]	–	–	–	–
Genetic algorithms	–	–	v [1,98]	–	
Fuzzy-based systems	–	–	–	–	–

"+" indicates good performance, "v" indicates average performance, "–" indicates poor performance, and " " indicates unknown performance

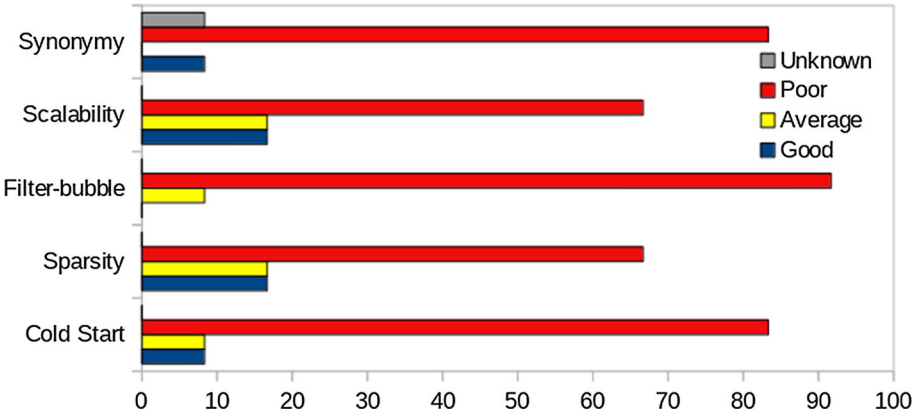


Fig. 4 The percentage of RS categories that perform good, average, poor, and with unknown performance over the five open issues studied in this research

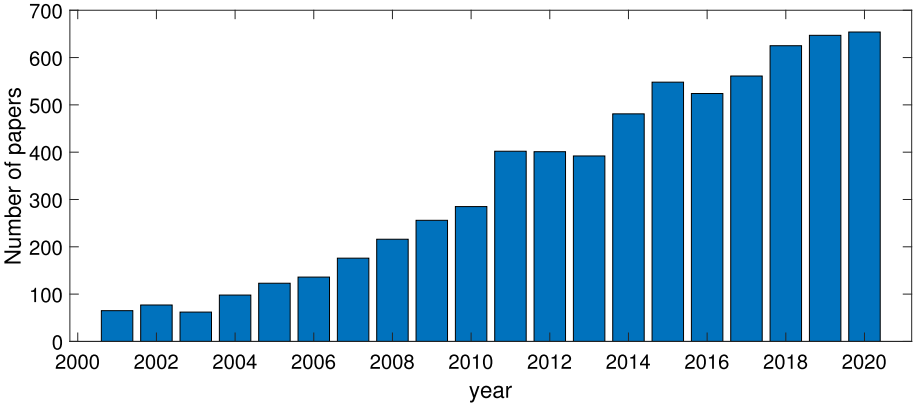


Fig. 5 Number of publications containing the phrase “recommender systems” in their title over the last 20 years. The data has been collected from Google Scholar

prevalent along with the Sparsity problem. The aforementioned conclusions were expected, in the sense that most CF-based approaches are vulnerable to a sparse dataset. In addition, the nature of the recommendation problem itself makes it prone to filter bubble, requiring additional mechanisms to alleviate this, often at the cost of accuracy.

Figure 5 depicts the number of publications in scientific journals and conferences containing the phrase "recommender systems" in their title over the last 20 years. It holds that their number dramatically increased by about ten times over the last twenty years, showing that the scientific topic of recommender systems belongs on the state-of-art topics of artificial intelligence.

Figure 6 shows the number of publications containing the phrase "recommender systems" and the corresponding collaborative filtering RS category in their text (instead of only in the title) during the last 20 years. It holds that the most popular category clearly is the neural network with 18,400 publications. The second, third, and fourth most popular categories are the association rules, dimensionality reduction, and similarity-based, respectively, with similar number of publications in the range of 8000–8500.

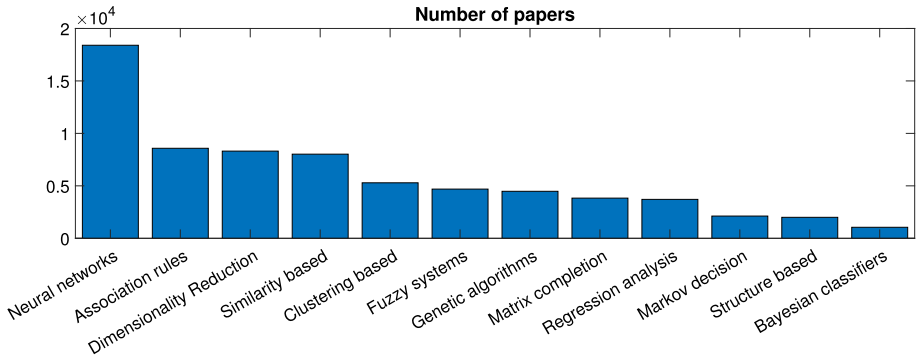


Fig. 6 Number of publications containing the phrase "recommender systems" and the corresponding collaborative filtering RS category in the text of the paper during the last 20 years. The data has been collected from Google Scholar

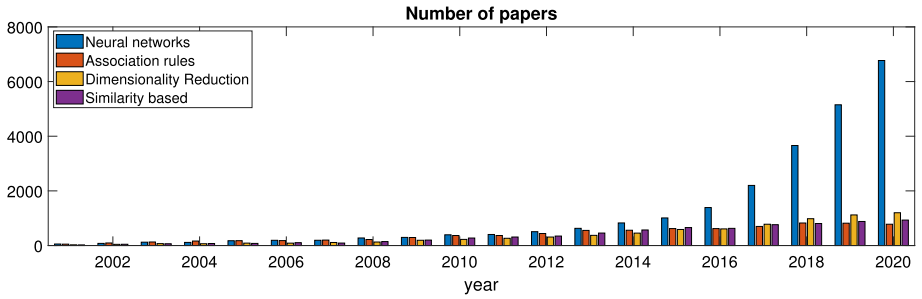


Fig. 7 Number of publications per year containing the phrase "recommender systems" and the corresponding collaborative filtering RS category in the text of the paper. The data has been collected from Google Scholar

Figure 7 shows the number of publications per year containing the phrase "recommender systems" and the four most popular collaborative filtering RS categories according to the results of Fig. 6. It holds that the most popular category, that is neural network, after 2014 shows exponential growth on the number of publications, approaching the number of 7000 publications in 2020. The growth in the number of papers for the other three collaborative filtering RS categories (association rules, dimensionality reduction, and similarity-based) appears to be constant, approaching the number of 1000 publications in 2020. It should be noticed that in 2001 the number of papers in the neural network category was only 23, the lowest number over the four categories, since each of the other three collaborative filtering RS categories (association rules, dimensionality reduction, and similarity-based) had about 30–60 papers. The data of Figs. 5, 6, and 7 has been collected from Google Scholar.

5 Conclusions

In this work, we presented a review of the recommendation approaches proposed in the entire research area of collaborative filtering recommender systems, in order to let the reader acquire a quick and complete understanding of this research area. We focused on the field of collaborative filtering systems, since it is reported in the literature to offer several advantages, such as agnosticity (no domain specific knowledge required for its use), can capture inherent

latent characteristics not specified in context-based systems, can provide more diverse recommendations (although still not to an acceptable enough degree), and generally presents more scalability. More specifically, to facilitate understanding, we provided a collaborative filtering recommender systems taxonomy, i.e., a categorization of each approach based on the tools and techniques employed. For each such category, we presented an overview of its most representative systems, while avoiding to present most of their implementation details. We remark that this study either (briefly) presented or included references to i) well-known papers, as well as, ii) the most representative of the recently published papers in each category, according to our knowledge. Hybrid systems, which appear to be the current trend in recent research, were assigned to a single category depending on their collaborative filtering core mechanism. Finally, we provided a comparison of collaborative filtering recommender system categories according to their ability to efficiently handle some of the well-known RS difficulties. In general, one can observe the plethora of different techniques employed in the context of collaborative filtering recommender systems (as well as recommender systems in general). We believe this diversity to be a hindrance in the ongoing research in the field, with this fact being the main motivation behind this survey. We believe strongly however, that a thorough quantitative analysis is also required, with this being the goal of future work, following up this survey. We plan to perform specific, detailed experiments to evaluate well-known collaborative filtering systems on the open issues described in Sect. 4. We hope that the present work will be useful to the RS research community, given that RS research is a multi-disciplinary field based on diverse techniques from various fields of Information Science. Thus, a careful organization of the available approaches is essential in order to obtain an overall view of the field. Finally, we identify several major trends which we believe will dominate the literature of this field in the years to come. Firstly, it is clear from the data presented in Sect. 4 that (deep) neural networks have come to dominate the field, and will continue to do in the future for some time. In addition, we believe that recommender systems will evolve far beyond the context of product recommendation, in more general aspects of everyday life, such as tourism, transportation, e-Health, and media [19]. As a result, the amount of data that those systems will be called upon to process will increase exponentially, leading to another important trend in future research, which will be their scalability and capability to process big data in order to provide recommendations (most probably in real time). Of course, research will also continue on classic open issues such as Serendipity; however, we believe the focus will move away from accuracy and RMSE minimization, an aspect which has dominated recommender systems research in the past.

Acknowledgements This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH–CREATE–INNOVATE (project code: T1EDK-02147).

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749. https://doi.org/10.1109/TKDE.2005.99
2. Agarwal D, Chen BC (2009) Regression-based latent factor models. In: 15th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '09. ACM, New York, pp 19–28
3. Aggarwal CC (2016) Recommender systems: the textbook, 1st edn. Springer, Berlin
4. Aggarwal CC et al (2016) Recommender systems. Springer, Berlin
5. Aggarwal CC, Gates SC, Yu PS (2004) On using partial supervision for text categorization. IEEE Trans Knowl Data Eng 16(2):245–255. https://doi.org/10.1109/TKDE.2004.1269601

6. Al-bashiri H, Abdulgabber MA, Romli A, Kahtan H (2018) An improved memory-based collaborative filtering method based on the topsis technique. PLoS ONE 13(10):1–26. https://doi.org/10.1371/journal.pone.0204434
7. Alhijawi B (2017) The use of the genetic algorithms in the recommender systems. Ph.D. thesis, Hashemite University
8. Al-Shamri MYH, Bharadwaj KK (2008) Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. Expert Syst Appl 35(3):1386–1399
9. Amatriain X, Jaimes A, Oliver N, Pujol JM (2011) Data mining methods for recommender systems. In: Recommender systems handbook. Springer, Berlin, pp 39–71
10. Bag S, Ghadge A, Tiwari MK (2019) An integrated recommender system for improved accuracy and aggregate diversity. Comput Ind Eng 130:187–197. https://doi.org/10.1016/j.cie.2019.02.028
11. Balabanović M, Shoham Y (1997) Fab: content-based, collaborative recommendation. Commun ACM 40(3):66–72. https://doi.org/10.1145/245108.245124
12. Batmaz Z, Yurekli A, Bilge A, Kaleli C (2019) A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 52(1):1–37
13. Bennett J, Lanning S, et al (2007) The netflix prize. In: Proceedings of KDD cup and workshop, vol. 2007, p. 35. New York, NY, USA
14. Berbague C, Karabadji NE, Seridi H (2018) Recommendation diversification using a weighted similarity measure in user based collaborative filtering. In: 2018 International symposium on programming and systems (ISPS), pp 1–6. https://doi.org/10.1109/ISPS.2018.8379011
15. Billsus D, Pazzani MJ (1998) Learning collaborative information filters. In: Proceedings of the fifteenth international conference on machine learning, ICML '98. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 46–54
16. Bobadilla J, Ortega F, Hernando A, Alcalá J (2011) Improving collaborative filtering recommender system results and performance using genetic algorithms. Knowl Based Syst 24:1310–1316
17. Bobadilla J, Ortega F, Hernando A, GutiéRrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132
18. Bojnordi E, Moradi P (2012) A novel collaborative filtering model based on combination of correlation method with matrix completion technique. In: Artificial intelligence and signal processing (AISP), 2012 16th CSI international symposium on, pp 191–194. IEEE
19. Bourke S (2015) The application of recommender systems in a multi site, multi domain environment. In: Proceedings of the 9th ACM conference on recommender systems, RecSys '15, p. 229. Association for Computing Machinery, New York. https://doi.org/10.1145/2792838.2799495
20. Brin S, Page L (1998) The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst 30(1–7):107–117. https://doi.org/10.1016/S0169-7552(98)00110-X
21. Candes EJ, Plan Y (2010) Matrix completion with noise. Proc IEEE 98(6):925–936
22. Chae DK, Lee SC, Lee SY, Kim SW (2018) On identifying k-nearest neighbors in neighborhood models for efficient and effective collaborative filtering. Neurocomputing 278:134–143. https://doi.org/10.1016/j.neucom.2017.06.081
23. Chiluka N, Andrade N, Pouwelse J (2011) A link prediction approach to recommendations in large-scale user-generated content systems. In: Proceedings of the 33rd European conference on advances in information retrieval, ECIR'11, pp 189–200. Springer, Berlin. http://dl.acm.org/citation.cfm?id=1996889.1996914
24. Comon P, Luciani X, De Almeida AL (2009) Tensor decompositions, alternating least squares and other tales. J Chemometrics: J Chemometrics Soc 23(7–8):393–405
25. Da'u A, Salim N (2019) Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intell Rev, pp 1–40. https://doi.org/10.1007/s10462-019-09744-1
26. Deng S, Huang L, Xu G, Wu X, Wu Z (2017) On deep learning for trust-aware recommendations in social networks. IEEE Trans Neural Netw Learn Syst 28(5):1164–1177. https://doi.org/10.1109/TNNLS.2016.2514368
27. Deshpande M, Karypis G (2004) Selective markov models for predicting web page accesses. ACM Trans Internet Technol 4(2):163–184
28. dos Santos C, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers, pp 69–78
29. Ebesu T, Fang Y (2017) Neural citation network for context-aware citation recommendation. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, pp 1093–1096. ACM
30. Ekstrand MD, Riedl JT, Konstan JA (2011) Collaborative filtering recommender systems. Found Trends Hum-Comput Interact 4(2):81–173. https://doi.org/10.1561/1100000009

31. Ge X, Liu J, Qi Q, Chen Z (2011) A new prediction approach based on linear regression for collaborative filtering. In: International conference on fuzzy systems and knowledge discovery, pp 2586–2590. IEEE
32. Han S, Mao H, Dally WJ (2015) Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149
33. Haveliwala TH (2003) Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. IEEE Trans Knowl Data Eng 15(4):784–796. https://doi.org/10.1109/TKDE.2003.1208999
34. Haveliwala T, Kamvar S, Jeh G (2003) An analytical comparison of approaches to personalizing pagerank. Technical Report 2003-35, Stanford InfoLab. http://ilpubs.stanford.edu:8090/596/
35. He X, Chua TS (2017) Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, pp 355–364. ACM
36. He X, Du X, Wang X, Tian F, Tang J, Chua TS (2018) Outer product-based neural collaborative filtering. arXiv preprint arXiv:1808.03912
37. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web. International World Wide Web Conferences Steering Committee, pp 173–182
38. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '99, pp 230–237. ACM, New York. https://doi.org/10.1145/312624.312682
39. Herlocker J, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf Retrieval 5(4):287–310. https://doi.org/10.1023/A:1020443909834
40. Hernando A, Bobadilla J, Ortega F (2016) A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowl-Based Syst 97:188–202
41. Hinton G, Deng L, Yu D, Dahl GE, Mohamed Ar, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97
42. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
43. Hu Y, Shi W, Li H, Hu X (2017) Mitigating data sparsity using similarity reinforcement-enhanced collaborative filtering. ACM Trans Internet Technol 17(3):3:11-3:120. https://doi.org/10.1145/3062179
44. Huang Z, Li X, Chen H (2005) Link prediction approach to collaborative filtering. In: Proceedings of the 5th ACM/IEEE-CS joint conference on digital libraries (JCDL '05), pp 141–142
45. Iaquinta L, De Gemmis M, Lops P, Semeraro G, Filannino M, Molino P (2008) Introducing serendipity in a content-based recommender system. In: 2008 Eighth international conference on hybrid intelligent systems, pp 168–173
46. Isinkaye F, Folajimi Y, Ojokoh B (2015) Recommendation systems: principles, methods and evaluation. Egypt Inform J 16(3):261–273
47. Jain AK (2010) Data clustering: 50 years beyond k-means. Pattern Recognit Lett 31(8):651–666
48. Jamali M, Ester M (2009) Trustwalker: a random walk model for combining trust-based and item-based recommendation. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '09, pp 397–406. ACM, New York. https://doi.org/10.1145/1557019.1557067
49. Jannach D, Resnick P, Tuzhilin A, Zanker M (2016) Recommender systems-beyond matrix completion. Commun ACM 59(11):94–102
50. Jeh G, Widom J (2002) Simrank: a measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '02, pp 538–543. ACM, New York. https://doi.org/10.1145/775047.775126
51. Jiang J, Li W, Dong A, Gou Q, Luo X (2020) A fast deep autoencoder for high-dimensional and sparse matrices in recommender systems. Neurocomputing 412:381–391
52. Kant V, Bharadwaj KK (2013) Integrating collaborative and reclusive methods for effective recommendations: a fuzzy Bayesian approach. Int J Intell Syst 28(11):1099–1123. https://doi.org/10.1002/int.21619
53. Khatri M (2012) A survey of naïve bayesian algorithms for similarity in recommendation systems. Int J Adv Res Comput Sci Softw Eng 2(5)
54. Khusro S, Ali Z, Ullah I (2016) Recommender systems: issues, challenges, and research opportunities. Springer, Singapore, pp 1179–1189
55. Khusro S, Ali Z, Ullah I (2016) Recommender systems: issues, challenges, and research opportunities. Springer, Singapore, pp 1179–1189. https://doi.org/10.1007/978-981-10-0557-2_112

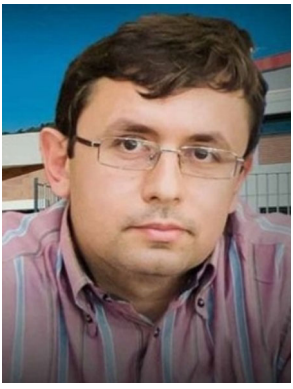
56. Kim Kj, Ahn H (2005) Using a clustering genetic algorithm to support customer segmentation for personalized recommender systems. In: Kim TG (ed) Artificial intelligence and simulation. Springer, Berlin, pp 409–415
57. Koohi H, Kiani K (2017) A new method to find neighbor users that improves the performance of collaborative filtering. Expert Syst Appl 83(C):30–39. https://doi.org/10.1016/j.eswa.2017.04.027
58. Koohi H, Kiani K (2016) User based collaborative filtering using fuzzy c-means. Measurement 91:134–139. https://doi.org/10.1016/j.measurement.2016.05.058
59. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 8:30–37
60. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
61. Kumar B, Sharma N (2016) Approaches, issues and challenges in recommender systems: a systematic review. Indian J Sci Technol **9**(47). http://www.indjst.org/index.php/indjst/article/view/94892
62. Leung CWK, Chan SCF, Chung Fl (2006) A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowl Inf Syst 10(3), 357–381. https://doi.org/10.1007/s10115-006-0002-1
63. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inform Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.20591
64. Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput 1:76–80
65. Liu H, Hu Z, Mian A, Tian H, Zhu X (2014) A new user similarity model to improve the accuracy of collaborative filtering. Knowl-Based Syst 56:156–166. https://doi.org/10.1016/j.knosys.2013.11.006
66. Liu X, Ouyang Y, Rong W, Xiong Z (2015) Item category aware conditional restricted Boltzmann machine based recommendation. In: International conference on neural information processing. Springer, pp 609–616
67. Li J, Zhang L, Meng F, Li F (2014) Recommendation algorithm based on link prediction and domain knowledge in retail transactions. Procedia Comput Sci 31:875–881. https://doi.org/10.1016/j.procs.2014.05.339
68. Mild A, Natter M (2002) Collaborative filtering or regression models for internet recommendation systems? J Target Meas Anal Mark 10(4):304–313
69. Moradi P, Ahmadian S, Akhlaghian F (2015) An effective trust-based recommendation method using a novel graph clustering algorithm. Physica A 436:462–481
70. Nguyen TT, Hui PM, Harper FM, Terveen L, Konstan JA (2014) Exploring the filter bubble: the effect of using recommender systems on content diversity. In: Proceedings of the 23rd international conference on world wide web, WWW '14. ACM, New York, pp 677–686
71. Nilashi M, Ibrahim O, Bagherifard K (2018) A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst Appl 92:507–520
72. Page L, Brin S, Motwani R, Winograd T (1998) The pagerank citation ranking: bringing order to the web. In: Proceedings of the 7th international world wide web conference. Brisbane, Australia, pp 161–172. https://www.citeseer.nj.nec.com/page98pagerank.html
73. Panagiotakis C (2015) Point clustering via voting maximization. J Classif 32(2):212–240
74. Panagiotakis C, Papadakis H, Grinias E, Komodakis N, Fragopoulou P, Tziritas G (2013) Interactive image segmentation based on synthetic graph coordinates. Pattern Recognit 46(11):2940–2952
75. Panagiotakis C, Papadakis H, Fragopoulou P (2018) Detection of hurriedly created abnormal profiles in recommender systems. In: International conference on intelligent systems
76. Panagiotakis C, Papadakis H, Fragopoulou P (2020) A user training error based correction approach combined with the synthetic coordinate recommender system. In: International conference on user modeling, adaptation and personalization
77. Panagiotakis C, Papadakis H, Fragopoulou P (2020) Personalized video summarization based exclusively on user preferences. In: European conference on information retrieval
78. Panagiotakis C, Papadakis H, Fragopoulou P (2020) Unsupervised and supervised methods for the detection of hurriedly created profiles in recommender systems. Mach Learn Cybern
79. Papadakis H, Panagiotakis C, Fragopoulou P (2014) Distributed detection of communities in complex networks using synthetic coordinates. J Stat Mech: Theory Exp 2014(3):P03013
80. Papadakis H, Panagiotakis C, Fragopoulou P (2017) Scor: a synthetic coordinate based recommender system. Expert Syst Appl 79:8–19
81. Park ST, Chu W (2009) Pairwise preference regression for cold-start recommendation. In: RecSys, pp 21–28
82. Perera D, Zimmermann R (2018) Lstm networks for online cross-network recommendations. In: IJCAI, pp 3825–3833

83. Ramezani M, Moradi P, Akhlaghian F (2014) A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains. Phys A: Stat Mech Appl 408:72–84. https://doi.org/10.1016/j.physa.2014.04.002
84. Ray S (2015) 7 types of regression techniques you should know! www.analyticsvidhya.com . https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/
85. Rencher ACW (2012) Methods of multivariate analysis. In: Wiley series in probability and statistics, chap. 10.1. Wiley, London
86. Rendle S (2012) Factorization machines with libfm. ACM Trans Intell Syst Technol 3(3):1–22
87. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning, pp 791–798. ACM
88. Salehi M, Pourzaferani M, Razavi SA (2013) Hybrid attribute-based recommender system for learning material using genetic algorithm and a multidimensional information model. Egypt Inform J 14(1):67–78
89. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, pp 285–295. ACM
90. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web, WWW '01, pp 285–295. ACM, New York. https://doi.org/10.1145/371920.372071
91. Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on world wide web, WWW '01, pp. 285–295. ACM, New York
92. Schein AI, Popescul A, Ungar LH, Pennock DM (2002) Methods and metrics for cold-start recommendations. In: Järvelin K, Beaulieu M, Baeza-Yates RA, Myaeng S (eds) SIGIR 2002: proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval, August 11–15, Tampere, Finland, pp 253–260. ACM. https://doi.org/10.1145/564376.564421
93. Shahabi C, Chen YS (2003) Web information personalization: challenges and approaches. In: Bianchi-Berthouze N (ed) Databases in networked information systems. Springer, Berlin, pp 5–15
94. Shah L, Hetal G, Prem B (2016) Survey on recommendation system. System 137(7)
95. Shani G, Heckerman D, Brafman RI (2005) An mdp-based recommender system. J Mach Learn Res 6:1265–1295
96. Shardanand U, Maes P (1995) Social information filtering: algorithms for automating "word of mouth". In: Proceedings of the SIGCHI conference on human factors in computing systems, CHI '95, pp 210–217. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. https://doi.org/10.1145/223904.223931
97. Sharif MA, Raghavan VV (2017) Link prediction based hybrid recommendation system using user-page preference graphs. In: Proceedings of the 2017 international joint conference on neural networks (IJCNN) . https://doi.org/10.1109/IJCNN.2017.7965981
98. Sharma L, Gera A (2013) A survey of recommendation system research challenges. Int J Eng Trends Technol
99. Sharma L, Gera A (2013) A survey of recommendation system: research challenges. Int J Eng Trends Technol
100. Shih HS, Shyur HJ, Lee ES (2007) An extension of topsis for group decision making. Math Comput Model 45(7):801–813. https://doi.org/10.1016/j.mcm.2006.03.023
101. Singh S, Bag S, Jenamani M (2015) Relative similarity based approach for improving aggregate recommendation diversity. In: 2015 Annual IEEE India conference (INDICON), pp 1–6. https://doi.org/10.1109/INDICON.2015.7443856
102. Smirnov A, Ponomarev A, Kashevnik A (2017) Multi-model service for recommending tourist attractions. In: Hammoudi S, Maciaszek LA, Missikoff MM, Camp O, Cordeiro J (eds) Enterprise information systems. Springer, Cham, pp 364–386
103. Son LH (2014) Hu-fcf: a hybrid user-based fuzzy collaborative filtering method in recommender systems. Expert Syst Appl 41(15):6861–6870. https://doi.org/10.1016/j.eswa.2014.05.001
104. Sorzano COS, Vargas J, Montano AP (2014) A survey of dimensionality reduction techniques. arXiv preprint arXiv:1403.2877
105. Strub F, Mary J, Gaudel R (2016) Hybrid collaborative filtering with autoencoders. arXiv preprint arXiv:1603.00806
106. Suzuki Y, Ozaki T (2017) Stacked denoising autoencoder-based deep collaborative filtering using the change of similarity. In: 2017 31st International conference on advanced information networking and applications workshops (WAINA), pp 498–502. IEEE

107. Tsai MH, Aggarwal C, Huang T (2014) Ranking in heterogeneous social media. In: Proceedings of the 7th ACM international conference on web search and data mining, WSDM '14. ACM, New York, pp 613–622. https://doi.org/10.1145/2556195.2556254
108. Tsai CF, Hung C (2012) Cluster ensembles in collaborative filtering recommendation. Appl Soft Comput 12(4):1417–1425
109. Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. In: Advances in neural information processing systems, pp 2643–2651
110. Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the fifth ACM conference on recommender systems, RecSys '11. ACM, New York, pp 109–116. https://doi.org/10.1145/2043932.2043955
111. Vucetic S, Obradovic Z (2005) Collaborative filtering using a regression-based approach. Knowl Inf Syst 7:1–22
112. Wang D, Zhang X, Yu D, Xu G, Deng S (2021) Came: content- and context-aware music embedding for recommendation. IEEE Trans Neural Netw Learn Syst 32(3):1375–1388. https://doi.org/10.1109/TNNLS.2020.2984665
113. Wu X, Cheng B, Chen J (2017) Collaborative filtering service recommendation based on a novel similarity computation method. IEEE Trans Serv Comput 10(3):352–365. https://doi.org/10.1109/TSC.2015.2479228
114. Wu X, Huang Y (2017) Sigra: a new similarity computation method in recommendation system. In: 2017 International conference on cyber-enabled distributed computing and knowledge discovery (CyberC), pp 148–154. https://doi.org/10.1109/CyberC.2017.89
115. Wu S, Ren W, Yu C, Chen G, Zhang D, Zhu J (2016) Personal recommendation using deep recurrent neural networks in netease. In: Data Engineering (ICDE), 2016 IEEE 32nd international conference on, pp 1218–1229. IEEE
116. Xie F, Chen Z, Shang J, Feng X, Li J (2015) A link prediction approach for item recommendation with complex number. Knowl-Based Syst 81:148–158. https://doi.org/10.1016/j.knosys.2015.02.013
117. Xie W, Ouyang Y, Ouyang J, Rong W, Xiong Z (2016) User occupation aware conditional restricted boltzmann machine based recommendation. In: Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE international conference on, pp 454–461. IEEE
118. Xi D, Zhuang F, Song B, Zhu Y, Chen S, Hong D, Chen T, Gu X, He Q (2020) Neural hierarchical factorization machines for user's event sequence analysis. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 1893–1896
119. Xue GR, Lin C, Yang Q, Xi W, Zeng HJ, Yu Y, Chen Z (2005) Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pp 114–121. ACM
120. Zenebea A, Norciob AF (2003) Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems. Fuzzy Sets Syst 160:76–94
121. Zhang Y, Koren J (2007) Efficient bayesian hierarchical user modeling for recommendation system. In: International ACM SIGIR conference on research and development in information retrieval
122. Zhang Z, Lin H, Liu K, Wu D, Zhang G, Lu J (2013) A hybrid fuzzy-based personalized recommender system for telecom products/services. Inf Sci 235:117–129. https://doi.org/10.1016/j.ins.2013.01.025
123. Zhang Z, Robinson D, Tepper J (2018) Detecting hate speech on twitter using a convolution-gru based deep neural network. In: European semantic web conference, pp 745–760. Springer
124. Zhang Q, Wang J, Huang H, Huang X, Gong Y (2017) Hashtag recommendation for multimodal microblog using co-attention network. In: Proceedings of the twenty-sixth international joint conference on artificial intelligence, IJCAI 2017, Melbourne, Australia, pp 3420–3426
125. Zhang S, Yao L, Sun A (2017) Deep learning based recommender system: a survey and new perspectives. arXiv preprint arXiv:1707.07435
126. Zheng Z, Ma H, Lyu MR, King I (2011) Qos-aware web service recommendation by collaborative filtering. IEEE Trans Serv Comput 4(2):140–152. https://doi.org/10.1109/TSC.2010.52
127. Zheng Z, Ma H, Lyu MR, King I (2009) Wsrec: a collaborative filtering based web service recommender system. In: 2009 IEEE international conference on web services, pp 437–444. https://doi.org/10.1109/ICWS.2009.30

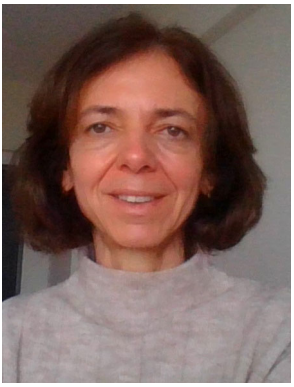
Harris Papadakis is an Assistant Professor of the Department of Electrical and Informatics Engineering of the Hellenic Mediterranean University in the field of "Distributed Services and Networks". He holds a PhD from the Department of Computer Science of the University of Crete, a degree in Computer Science from the Department of Computer Science of the University of Crete and a postgraduate MSc diploma from the Department of Computer Engineering of the University of Patras. He has participated as researcher and head in several projects. He has more than 500 citations with an h-index of 8 and a i10-index of 7. His research interests include: Parallel and Distributed Systems, Peer Systems, Large-Scale Distributed Systems, Computer Networks, Recommender Systems and Graph Analysis.

Antonis Papagrigoriou received his BSc Diploma from the Department of Electrical and Computer Engineering, of Democritus University of Thrace (Greece) in 2000 and his MSc in Telecommunications from the same Department in 2002. He is working as a Senior Software Engineer in the OSS Department of NOVA, a private Greek telecommunication company, since 2004. From 2011, he has been involved in multiple European and National research projects as external Research Assistant at the Technological Educational Institute of Crete and Hellenic Mediterranean University. Areas of Interest: Software Developing, Research on Embedded Systems, Research on Recommended Systems.



Costas Panagiotakis was born in Heraklion, Crete, Greece, in 1979. He received the B.Sc. (9.0/10 highest honors), M.Sc. (highest honors) and Ph.D. degrees in Computer Science from University of Crete in 2001, 2003 and 2007, respectively. Currently, she is Associate Professor and Head in Department of Management Science and Technology, Hellenic Mediterranean University and Director of Data Science, Multimedia and Modelling Laboratory. He is also a researcher at the Computational Vision and Robotics Laboratory, Institute of Computer Science, Foundation for Research and Technology-Hellas. He is the author of one book (monograph) and more than 80 articles in international journals and conferences. His research interests include signal processing, image and video analysis, 3D animation, multimedia and pattern recognition. [homepage: https://sites.google.com/site/costaspanagiotakis/]

Eleftherios Kosmas received his BSc and MSc degrees in Computer Science from the University of Ioannina in 2005 and 2008, and his PhD degree in Computer Science from the University of Crete in 2015. Currently, he is a Contract Lecturer at the Department of Electrical and Computer Engineering of the Hellenic Mediterranean University and a Postdoc Researcher at the Department of Computer Science of the University of Crete. He graduated first in his BSc class and with the best mark in the history of the Department. He received several distinctions and prizes, including scholarships from the Greek Government for excellent performance. He has acquired two research grants as Principal Investigator and participated in two other research projects. He is the co-author of two book chapters and has publications in top rated conferences and journals. His research interests include principles of parallel and distributed computing, and programming of parallel and distributed systems.



Paraskevi Fragopoulou received her BSc in Computer Science from the University of Crete in 1989, and the MSc and PhD degrees in Computer Science from Queen's University, Ontario, Canada, in 1990 and 1995, respectively. Currently, he is also a Professor at the Department Electrical and Computer Engineering of the Hellenic Mediterranean University. She is also a Collaborating Researcher at the Institute of Computer Science, Foundation for Research and Technology-Hellas (FORTH-ICS), as member of the Distributed Computing Systems and Cybersecurity Laboratory. Prof. Fragopoulou has co-authored more than 60 conference/journal papers and book chapters. Her primary research interest is in the areas of Internet Safety and Internet Technology, Security of Computer Networks and Communications, Distributed and Parallel Computing, Peer-to-Peer systems and Distributed Computing. She has participated often as a Principal Invesigator in several projects funded by the European Commission and by the Greek Government.