Chapter 2
Review of Previous Work Related to Recommender Systems

Abstract The large amount of information resources that are available to users imposes new requirements on the software systems that handle the information. This chapter reviews the state of the art of the main approaches to designing RSs that address the problems caused by information overload. In general, the methods implemented in a RS fall within one of the following categories: (a) Content-based Methods, (b) Collaborative Methods and (c) Hybrid Methods.

2.1 Content-Based Methods

Modern information systems embed the ability to monitor and analyze users' actions to determine the best way to interact with them. Ideally, each users actions are logged separately and analyzed to generate an individual user profile. All the information about a user, extracted either by monitoring user actions or by examining the objects the user has evaluated [9], is stored and utilized to customize services offered. This user modeling approach is known as *content-based learning*. The main assumption behind it is that a user's behavior remains unchanged through time; therefore, the content of past user actions may be used to predict the desired content of their future actions [4, 27]. Therefore, in content-based recommendation methods, the rating $R(u, i)$ of the item i for the user u is typically estimated based on ratings assigned by user u to the items $I_n \in I$ that are "similar" to item i in terms of their content, as defined by their associated features.

To be able to search through a collection of items and make observations about the similarity between objects that are not directly comparable, we must transform raw data at a certain level of information granularity. Information granules refer to a collection of data that contain only essential information. Such granulation allows more efficient processing for extracting features and computing numerical representations that characterize an item. As a result, the large amount of detailed information of one item is reduced to a limited set of features. Each feature is a vector of low dimensionality, which captures some aspects of the item and can be used to determine item similarity. Therefore, an item i could be described by a feature vector

$$F(i) = [\, feature_1(i), feature_2(i), feature_3(i), \ldots feature_n(i)]. \tag{2.1}$$

A.S. Lampropoulos and G.A. Tsihrintzis, *Machine Learning Paradigms*,
Intelligent Systems Reference Library 92, DOI 10.1007/978-3-319-19135-5_2

For example, in a music recommendation application, in order to recommend music files to user u, the content-based RS attempts to build a profile of the user's preferences based on features presented in music files that the user u has rated with high rating degrees. Consequently, only music files that have a high degree of similarity with these highly rated files would be recommended to the user. This method is known as "item-to-item correlation" [41]. The type of user profile derived by a content-based RS depends on the learning method which is utilized by the system. This approach to the recommendation process has its roots in information retrieval and information filtering [3, 36]. Retrieval-based approaches utilize interactive learning techniques such as *relevance feedback* methods, in order to organize and retrieve data in an effective personalized way. In relevance feedback methods, the user is part of the item-management process, which means that the user evaluates the results provided by the system. Then, the system adapts, its performance according to the user's preferences. In this way, the method of relevance feedback has the efficiency not only to take into account the user subjectivity in perceiving the content of items, but also to eliminate the gap between high-level semantics and low-level features which are usually used for the content description of items [12, 13, 35].

Besides the heuristics that are based mostly on information retrieval methods [3, 12, 13, 35, 36] such as the Rocchio algorithm or correlation-based schemes, other techniques for content-based recommendation utilize Pattern Recognition/Machine Learning approaches, such as Bayesian classifiers [28], clustering methods, decision trees, and artificial neural networks.

These techniques differ from information retrieval-based approaches as they calculate utility predictions based not on a heuristic formula, such as a cosine similarity measure, but rather are based on a model learnt from the underlying data using statistical and machine learning techniques. For example, based on a set of Web pages that were rated by the user as "relevant" or "irrelevant," the naive Bayesian classifier is used in [28] to classify unrated Web pages.

Some examples of content-based methods come from the area of music data. In [10, 19, 24, 25, 47], they recommend pieces that are similar to users' favorites in terms of music content such as mood and rhythm. This allows a rich artist variety and various pieces, including unrated ones, to be recommended. To achieve this, it is necessary to associate user preferences with music content by using a practical database where most users tend to rate few pieces as favorites.

A relevance feedback approach for music recommendation was presented in [19] and based on the TreeQ vector quantization process initially proposed by Foote [14]. More specifically, relevance feedback was incorporated into the **user model** by modifying the quantization weights of desired vectors. Also, a relevance feedback music retrieval system, based on SVM Active Learning, was presented in [25], which retrieves the desired music piece according to mood and style similarity.

In [2], the authors explore the relation between the user's rating input, musical pieces with high degree of rating that were defined as the listener's favorite music, and music features. Specifically, labeled music pieces from specific artists were analyzed in order to build a correlation between user ratings and artists through music features. Their system forms the user profile as preference for music pieces of a specific artist.

They confirmed that favorite music pieces were concentrated along certain music features.

The system in [52] proposes the development of a user-driven similarity function by combining timbre-, tempo-, genre-, mood-, and year-related features into the overall similarity function. More specifically, similarity is based on a weighted combination of these features and the end-user can specify his/her personal definition of similarity by weighting them.

The work in [15] tries to extend the use of signal approximation and characterization from genre classification to recognition of user taste. The idea is to learn music preferences by applying instance-based classifiers to user profiles. In other words, this system does *not* build an individual profile for every user, but instead tries to recognize his/her favorite genre by applying instance-based classifiers to user rating preferences by his/her music playlist.

2.2 Collaborative Methods

CF methods are based on the assumption that similar users prefer similar items or that a user expresses similar preferences for similar items. Instead of performing content indexing or content analysis, CF systems rely entirely on interest ratings from the members of a participating community [18]. CF methods are categorized into two general classes, namely *model-based* and *memory-based* [1, 7].

Model-based algorithms use the underlying data to learn a probabilistic model, such as a cluster model or a Bayesian network model [7, 53], using statistical and machine learning techniques. Subsequently, they use the model to make predictions. The clustering model [5, 51] works by clustering similar users in the same class and estimating the probability that a particular user is in a particular class. From there, the clustering model computes the conditional probability of ratings.

Memory-based methods, store raw preference information in computer memory and access it as needed to find similar users or items and to make predictions. In [29], CF was formulated as a classification problem. Specifically, based on a set of user ratings about items, they try to induce a model for each user that would allow the classification of unseen items into two or more classes, each of which corresponds to different points in the accepted rating scale.

Memory-based CF methods can be further divided into two groups, namely user-based and item-based [37] methods. On the one hand, user-based methods look for users (also called "neighbors") similar to the active user and calculate a predicted rating as a weighted average of the neighbor's ratings on the desired item. On the other hand, item-based methods look for similar items for an active user.

2.2.1 User-Based Collaborative Filtering Systems

User-based CF systems are systems that utilize **memory-based algorithms**, meaning that they operate over the entire user-item matrix R, to make predictions. The majority

of such systems mainly deal with **user-user similarity calculations**, meaning that they utilize user neighborhoods, constructed as collections of similar users. In other words, they deal with the rows of the user-item matrix, R, in order to generate their results. For example, in a personalized music RS called RINGO [43], similarities between the tastes of different users are utilized to recommend music items. This user-based CF approach works as follows: A new user is matched against the database to discover neighbors, which are other customers who, in the past, have had a similar taste as the new user, i.e. who have bought similar items as the new user. Items (unknown to the new user) that these neighbors like are then recommended to the new user. The main steps of this process are:

1. Representation of Input data,
2. Neighborhood Formation, and
3. Recommendation Generation.

2.2.1.1 Representation of Input Data

To represent input data, one needs to define a set of ratings of users into a user-item matrix, R, where each $R(u, i)$ represents the rating value assigned by the user u to the item i. As users are not obligated to provide their opinion for all items, the resulting user-item matrix may be a **sparse matrix**. This sparsity of the user-item matrix is the main reason causing filtering algorithms not to produce satisfactory results. Therefore, a number of techniques were proposed to reduce the sparsity of the initial user-item matrix to improve the efficiency of the RS. *Default Voting* is the simplest technique used to reduce sparsity. A default rating value is inserted to items for which there does not exist a rating value. This rating value is selected to be neutral or somewhat indicative of negative preferences for unseen items [7].

An extension of the method of Default Voting is to use either the *User Average Scheme* or the *Item Average Scheme* or the *Composite Scheme* [39]. More specifically:

- In the *User Average Scheme*, for each user, u, the average user rating over all the items is computed, $\overline{R}(u)$. This is expressed as the average of the corresponding row in the user-item matrix. The user average is then used to replace any missing $R(u, i)$ value. This approach is based on the idea that a user's rating for a new item could be simply predicted if we take into account the same user's past ratings.
- In the *Item Average Scheme*, for each item, the item average over all users is computed, $\overline{R}(i)$. This is expressed as the average of the corresponding column in the user-item matrix. The item average is then used as a fill-in for missing values $R(u, i)$ in the matrix.
- In the *Composite Scheme*, the collected information for items and users both contribute to the final result. The main idea behind this method is to use the average of user u on item i as a base prediction and then add a correction term to it based on how the specific item was rated by other users.

The scheme works as follows: When a missing entry regarding the rating of user u on item i is located, initially, the user average $\overline{R}(u)$ is calculated as the average of the corresponding user-item matrix row. Then, we search for existing ratings in the column which correspond to item i. Assuming that a set of l users, $U = \{u_1, u_2, \ldots, u_l\}$, has provided a rating for item i, we can compute a correction term for each user $u \in L$ equal to $\delta_k = R(u_k, i) - \overline{R}(u_k)$. After the corrections for all users in U are computed, the composite rating can be calculated as:

$$R(u,i) = \begin{cases} \overline{R}(u) + \frac{\sum_{k=1}^{l} \delta_k}{l}, & \text{if user } u \text{ has not rated item } i \\ R, & \text{if user } u \text{ has rated item } i \text{ with } R. \end{cases} \tag{2.2}$$

An alternative way of utilizing the composite scheme is through a simple transposition: first compute the item average, $\overline{R}(i_k)$, (i.e., average of the column which corresponds to item i) and then compute the correction terms, δ_k, by scanning through all l items $I = \{i_1, i_2, \ldots, i_l\}$ rated by user k. The fill-in value of $R(u, i)$ would then be:

$$R(u,i) = \overline{R}(i) + \frac{\sum_{k=1}^{l} \delta_k}{l}, \tag{2.3}$$

where l is the count of items rated by user u and the correction terms are computed for all items in I as $\delta_k = R(u, i_k) - \overline{R}(i_k)$

After generating a reduced-dimensionality matrix, we could use a vector similarity metric to compute the proximity between users and hence to form *neighborhoods of users* [38], as discussed in the following.

2.2.1.2 Neighborhood Formation

In this step of the recommendation process, the *similarity* between users is calculated in the user-item matrix, R, i.e., users similar to the active user, u_a, form a proximity-based neighborhood with him. More specifically, neighborhood formation is implemented in two steps: Initially, the similarity between all the users in the user-item matrix, R, is calculated with the help of some proximity metrics. The second step is the actual neighborhood generation for the active user, where the similarities of users are processed in order to select those users that will constitute the neighborhood of the active user. To find the similarity between users u_a and u_b, we can utilize the *Pearson correlation metric*. The Pearson correlation was initially introduced in the context of the GroupLens project [33, 43], as follows: Let us assume that a set of m users u_k, where $k = 1, 2, \ldots, m$, $U_m = \{u_1, u_2, \ldots, u_m\}$, have provided a rating $R(u_k, i_l)$ for item i_l, where $l = 1, 2, \ldots, n$, $I_n = \{i_1, i_2, \ldots, i_n\}$ is the set of items. The Pearson correlation coefficient is given by:

$$sim(u_a, u_b) = \frac{\sum_{l=1}^{n} (R(u_a, i_l) - \overline{R}(u_a))(R(u_b, i_l) - \overline{R}(u_b))}{\sqrt{\sum_{l=1}^{n} (R(u_a, i_l) - \overline{R}(u_a))^2 \sum_{l=1}^{n} (R(u_b, i_l) - \overline{R}(u_b))^2}}. \quad (2.4)$$

Another metric similarity uses the *cosine-based approach* [7], according to which the two users u_a and u_b, are considered as two vectors in *n-dimensional* item-space, where $n = |I_n|$. The similarity between two vectors can be measured by computing the cosine angle between them:

$$sim(u_a, u_b) = \cos(\overrightarrow{u_a}, \overrightarrow{u_b}) = \frac{\sum_{l=1}^{n} R(u_a, i_l) R(u_b, i_l)}{\sqrt{\sum_{l=1}^{n} R(u_a, i_l)^2} \sqrt{\sum_{l=1}^{n} R(u_b, i_l)^2}}. \quad (2.5)$$

In RS, the use of the Pearson correlation similarity metric to estimate the proximity among users performs better than the cosine similarity [7].

At this point in the recommendation process, a single user is selected who is called the *active user*. The active user is the user for whom the RS will produce predictions and proceed with generating his/her neighborhood of users. A *similarity matrix* S is generated, containing the similarity values between all users. For example, the ith row in the similarity matrix represents the similarity between user u_i and all the other users. Therefore, from this similarity matrix S various schemes can be used in order to select the users that are most similar to the active user. One such scheme is the *center-based scheme*, in which from the row of the active user u_a are selected those users who have the highest similarity value with the active user.

Another scheme for neighborhood formation is the *aggregate neighborhood formation scheme*. In this scheme, a neighborhood of users is created by finding users who are closest to the *centroid* of the current neighborhood and not by finding the users who are closest to the active user himself/herself. This scheme allows all users to take part in the formation of the neighborhood, as they are gradually selected and added to it.

2.2.1.3 Generation of Recommendations

The generation of recommendations is represented by predicting a rating, i.e., by computing a numerical value which constitutes a predicted opinion of the active user u_a for an item i_j unseen by him/her. This predicted value should be within the same accepted numerical scale as the other ratings in the initial user-item matrix R. In the generation of predictions, only those users participate that lie within the neighborhood of the active user. In other words, only a subset of k users participate

from the m users in the set U_m that have provided ratings for the specific item i_j, $U_k \subseteq U_m$. Therefore, a *prediction score* P_{u_a,i_j} is computed as follows [33]:

$$P_{u_a,i_j} = \overline{R}(u_a) + \frac{\sum_{t=1}^{k} (R(u_t, i_j) - \overline{R}(u_t)) * sim(u_a, u_t)}{\sum_{t=1}^{k} |sim(u_a, u_t)|}, \quad \text{where } U_k \subseteq U_l \quad (2.6)$$

Here, $\overline{R}(u_a)$ and $\overline{R}(u_t)$ are the average rating of the active user u_a and u_t, respectively, while $R(u_t, i_j)$ is the rating given by user u_t to item i_j. Similarity $sim(u_a, u_t)$ is the similarity among users u_a and u_t, computed using the Pearson correlation in Eq. 2.4. Finally, the RS will output several items with the best predicted ratings as the recommendation list.

An alternative output of a RS is the *top-N recommendations* output. In this case, recommendations form a list of N items that the active user is expected to like the most. For the generation of this list, users are ranked first according to their similarity to the active user. The k most similar (i.e. most highly ranked) users are selected as the k-nearest neighbors of the active user u_a. The frequency count of an item is calculated by scanning the rating of the item by the k-nearest neighbors. Then, the items are sorted based on frequency count. The N most frequent items that have not been rated by the active user are selected as the top-N recommendations [23].

2.2.2 Item-Based Collaborative Filtering Systems

A different approach [20, 37] is based on item relations and not on user relations, as in classic CF. Since the relationships between users are relatively dynamic, as they continuously buy new products, it is computationally hard to calculate the user-to-user matrix online. This causes the user-based CF approach to be relatively expensive in terms of computational load. In the item-based CF algorithm, we look into the set of items, denoted by I_{u_a}, that the active user, u_a, has rated and compute how similar they are to the target item i_t. Then, we select the k most similar items $I_k = \{i_1, i_2, \ldots, i_k\}$, based on their corresponding similarities $\{sim(i_t, i_1), sim(i_t, i_2), \ldots, sim(i_t, i_k)\}$. The predictions can then be computed by taking a weighted average of the active user's ratings on these similar items. The main steps in this approach are the same as in user-based CF. The difference in the present approach is that instead of calculating similarities between two users who have provided ratings for a common item, we calculate similarities between two items i_t, i_j which have been rated by a common user u_a. Therefore, the Pearson correlation coefficient and cosine similarity are, respectively, given as:

$$sim(i_t, i_j) = \frac{\sum_{l=1}^{n} (R(u_l, i_t) - \overline{R}(i_t))(R(u_l, i_j) - \overline{R}(i_j))}{\sqrt{\sum_{l=1}^{n} (R(u_l, i_t) - \overline{R}(i_t))^2 \sum_{l=1}^{n} (R(u_l, i_j) - \overline{R}(i_j))^2}} \tag{2.7}$$

$$sim(i_t, i_j) = \cos(\overrightarrow{i_t}, \overrightarrow{i_j}) = \frac{\sum_{l=1}^{n} R(u_l, i_t) R(u_l, i_j)}{\sqrt{\sum_{l=1}^{n} R(u_l, i_t)^2} \sqrt{\sum_{l=1}^{n} R(u_l, i_j)^2}}. \tag{2.8}$$

Next, the similarities between all items in the initial user-item matrix, R, are calculated. The final step in the CF procedure is to isolate k items from n, ($I_k \subseteq I_n$) in order to share the greatest similarity with item i_t for which we are seeking a prediction, form its neighborhood of items, and proceed with prediction generation. A prediction on item i_t for active user u_a is computed as the sum of ratings given by the active user on items belonging to the neighborhood I_k. These ratings are weighted by the corresponding similarity, $sim(i_t, i_j)$ between item i_t and item i_j, with $j = 1, 2, \ldots, k$, taken from neighborhood I_k:

$$P_{u_a, i_j} = \frac{\sum_{j=1}^{k} sim(i_t, i_j) * R(u_a, i_j)}{\sum_{j=1}^{k} \left| sim(i_t, i_j) \right|} \text{ where } I_k \subseteq I_n. \tag{2.9}$$

In [16], the authors proposed that the long-term interest profile of a user (*task profile*) be established either by explicitly providing some items associated with the current task or by implicitly observing the user behavior (*intent*). By utilizing the item-to-item correlation matrix, items that resemble the items in the task profile are selected for recommendation. Since they match the task profile, these items fit the current task of the user. Before recommending them to the user, these items will be re-ranked to fit the user interests based on the interest prediction.

2.2.3 Personality Diagnosis

Personality diagnosis may be thought of as a hybrid between memory and model-based approaches of CF. The main characteristic is that predictions have meaningful probabilistic semantics. Moreover, this approach assumes that preferences constitute a characterization of their underlying personality type for each user. Therefore, taking into consideration the active user's known ratings of items, it is possible to estimate the probability that he/she has the same personality type with another user. The personality type of a given user is taken to be the vector of "true" ratings for items

the user has seen. A true rating differs from the actually reported rating given by a user by an amount of (Gaussian) noise. Given the personality type of a user, the personality diagnosis approach estimates the probability that the given user is of the same personality type as other users in the system, and, consequently, estimates the probability that the user will like some new item [30].

The personality type for each user u_k is formulated as follows, where $k = 1, 2, \ldots, m$, $U_m = \{u_1, u_2, \ldots, u_m\}$, and the user u_k has a number of preferred items in $I_n = \{i_1, i_2, \ldots, i_n\}$:

$$\overset{true}{R}(u_k) = \left\{ \overset{true}{R}(u_k, i_1), \overset{true}{R}(u_k, i_2), \ldots, \overset{true}{R}(u_k, i_n) \right\}. \tag{2.10}$$

Here, $\overset{true}{R}(u_k, i_l)$, with $i_l \in I_n$ and $l = 1, 2, \ldots, n$, stands for true rating by user u_k of the item i_l. It is important to note the difference between *true* and *reported* (given) ratings of the user. The true ratings encode the underlying internal preferences for a user that are *not* directly accessible by the designer of the RS. However, the reported ratings are those which were provided by users and utilized by the RS.

It is assumed that the reported ratings given by users include Gaussian noise. This assumption has the meaning that one user could report different ratings for the same items under different situations, depending on the context. Thus, we can assume that the rating reported by the user for an item i_l is drawn from an independent normal distribution with mean $\overset{true}{R}(u_k, i_l)$. Particularly:

$$Pr\left(R(u_k, i_l) = x \middle| \overset{true}{R}(u_k, i_l) = y \right) \propto e^{-\frac{(x-y)^2}{2\sigma^2}}, \tag{2.11}$$

where σ is a free parameter, x is the rating that the user has reported to the RS, and y is the true rating value that the user u_k would have reported if there no noise were present.

Furthermore, we assume that the distribution of personality types in the rating array R of users-items is representative of the personalities found in the target population of users. Therefore, taking into account this assumption, we can formulate the prior probability $Pr\left(\overset{true}{R}(u_a) = \upsilon\right)$ that the active user u_a rates items accordingly to a vector υ as given by the frequency that the other users rate according to υ. Thereby, instead of explicitly counting occurrences, we simply define $\overset{true}{R}(u_a)$ to be a random variable that can take one of m values, $(R(u_1), R(u_2), \ldots, R(u_m))$, each with probability $\frac{1}{m}$:

$$Pr\left(\overset{true}{R}(u_a) = R(u_k)\right) = \frac{1}{m}. \tag{2.12}$$

Combining Eqs. 2.11 and 2.12 and given the active user's ratings, we can compute the probability that the active user is of the same personality type as any other user,

by applying the Bayes rule:

$$\begin{aligned} & Pr\left(\overset{true}{R}(u_a) = R(u_k) | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right) \\ & \propto Pr\left(R(u_a, i_1) = x_1 | \overset{true}{R}(u_a, i_1) = R(u_a, i_1)\right) \\ & \ldots Pr\left(R(u_a, i_n) = x_n | \overset{true}{R}(u_a, i_n) = R(u_a, i_n)\right) \\ & \cdot Pr\left(\overset{true}{R}(u_a) = R(u_k)\right). \end{aligned} \tag{2.13}$$

Hence, computing this quantity for each user u_k, we can compute the probability distribution for the active user's rating of an unseen item i_j. This probability distribution corresponds to the prediction P_{u_a,i_j} produced by the RS and equals the expected rating value of active user u_a for the item i_j:

$$\begin{aligned} P_{u_a,i_j} &= Pr\left(R(u_a, i_j) = x_j | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right) \\ &= \sum_{k=1}^{m} \Pr\left(R(u_a, i_j) = x_j | \overset{true}{R}(u_a) = R(u_k)\right) \\ &\cdot \Pr\left(\overset{true}{R}(u_a) = R(u_k) | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\right). \end{aligned} \tag{2.14}$$

The model is depicted as a naive Bayesian network with the structure of a classical diagnostic model as follows:

- Firstly, we observe ratings and, using Eq. 2.13, compute the probability that each personality type is the cause. Ratings can be considered as "symptoms" while personality types as "diseases" leading to those symptoms in the diagnostic model.
- Secondly, we can compute the probability of rating values for an unseen item using Eq. 2.14. The most probable rating is returned as the prediction of the RS.

An alternative interpretation of personality diagnosis is to consider it as a clustering method with exactly one user per cluster. This is so because each user corresponds to a single personality type and the effort is to assign the active user to one of these clusters [7, 51].

An additional interpretation of personality diagnosis is that the active user is assumed to be "generated" by choosing one of the other users uniformly at random and adding Gaussian noise to his/her ratings. Given the active user's known ratings, we can infer the probability that he/she be actually one of other users and then compute probabilities for ratings of other items.

2.3 Hybrid Methods

Hybrid methods combine two or more recommendation techniques to achieve better performance and to take out drawbacks of each technique separately. Usually, CF

methods are combined with content-based methods. According to [1], hybrid RS could be classified into the following categories:

- Combining Separate Recommenders
- Adding Content-Based Characteristics to Collaborative Models
- Adding Collaborative Characteristics to Content-Based Models
- A Single Unifying Recommendation Model.

Combining Separate Recommenders

The Hybrid RS of this category include two separate systems, a collaborative one and a content-based one. There are four different ways of combining these two separate systems, namely the following:

- *Weighted Hybridization Method*. The outputs (ratings) acquired by individual RS are combined together to produce a single final recommendation using either a linear combination [11] or a voting scheme [29]. The *P-Tango* system [11] initially gives equal weights to both recommenders, but gradually adjusts the weights as predictions about user ratings are confirmed or not. The system keeps the two filtering approaches separate and this allows the benefit from individual advantages.
- *Switched Hybridization Method*. The system switches between recommendation techniques selecting the method that gives better recommendations for the current situation depending on some recommendation "quality" metric. A characteristic example of such a recommender is *The Daily Learner* [6], which selects the recommender sub-system that provides the higher level of confidence. Another example of this method is presented in [50] where either the content-based or the collaborative filtering technique is selected according to which of the two provided better consistency with past ratings of the user.
- *Mixed Hybridization Method*. In this method, the results from different recommender sub-systems are presented simultaneously. An example of such a recommender is given in [45] where they utilize a content-based technique based on textual descriptions of TV shows and collaborative information about users' preferences. Recommendations from both techniques are provided together in the final suggested program.
- *Cascade Hybridization Method*. In this method, one recommendation technique is utilized to produce a coarse ranking of candidates, while the second technique focuses only on those items for which additional refinement is needed. This method is more efficient than the weighted hybridization method which applies all of its techniques on all items. The computational burden of this hybrid approach is relatively small because recommendation candidates in the second level are partially eliminated in the first level. Moreover this method is more tolerant to noise in the operation of low-priority recommendations, since ratings of the high level recommender can only be refined, but never over-turned [9]. In other words, cascade hybridization methods can be analyzed into two sequential stages. The first stage (content-based method or knowledge-based/collaborative) selects intermediate recommendations. Then, the second stage (collaborative/content-based method

or knowledge-based) selects appropriate items from the recommendations of the first stage. Burke [8] developed a restaurant RS called *EntreeC*. The system first selects several restaurants that match a user's preferred cuisine (e.g., Italian, Chinese, etc.) with a knowledge-based method. In the knowledge-based method, the authors construct a feature vector according to defined attributes that characterize the restaurants. This method is similar to content-based methods; however, it must be noted that these metadata are content-independent and for this reason the term *knowledge-based* is utilized. These restaurants are then ranked with a collaborative method.

2.3.1 Adding Content-Based Characteristics to Collaborative Models

In [29], the authors proposed *collaboration via content*. This is a method that uses a prediction scheme similar to the standard CF, in which similarity among users is not computed on provided ratings, but rather on the content-based profile of each user. The underlying intuition is that like-minded users are likely to have similar content-based models and that this similarity relation can be detected without requiring overlapping ratings. The main limitation of this approach is that the similarity of users is computed using Pearson's correlation coefficient between content-based weight vectors.

On the other hand, in [26] the authors proposed the *content-boosted collaborative filtering* approach, which exploits a content-based predictor to enhance existing user data and then provides personalized suggestions through CF. The content-based predictor is applied to each row of the initial user-item matrix, corresponding to each user, and gradually generates a pseudo user-item matrix that is a full dense matrix. The similarity between the active user, u_a, and another user, u_i, is computed with CF using the new pseudo user-item matrix.

2.3.2 Adding Collaborative Characteristics to Content-Based Models

The main technique of this category is to apply dimensionality reduction on a group of content-based profiles. In [46], the authors used *latent semantic indexing* to create a collaborative view of a collection of user profiles represented as term vectors. This technique results in performance improvement in comparison with the pure content-based approach.

2.3.3 A Single Unifying Recommendation Model

A general unifying model that incorporates content-based and collaborative characteristics was proposed in [5], where the authors present the use of content-based and collaborative characteristics (e.g., the age or gender of users or the genre of movies) in a single rule-based classifier. Single unifying models were also presented in [31], where the authors utilized a unified probabilistic method for combining collaborative and content-based recommendations.

2.3.4 Other Types of Recommender Systems

Demographics-based RS. The basis for recommendations in demographics-based RS is the use of prior knowledge on demographic information about the users and their opinions for the recommended items. Demographics-based RS classify their users according to personal demographic data (e.g. age and gender) and classify items into user classes. Approaches falling into this group can be found in Grundy [34], a system for book recommendation, and in [21] for marketing recommendations. Similarly to CF, demographic techniques also employ user-to-user correlations, but differ in the fact that they do not require a history of user ratings. An additional example of a demographics-based RS is described in [29], in which information about users is taken from their home-pages to avoid the need to maintain a history of user ratings. Demographic characteristics for users (e.g. their age and gender) is also utilized in [5].

Knowledge-based RS. Knowledge-based RS use prior knowledge on how the recommended items fulfill the user needs. Thus, the goal of a knowledge-based RS is to reason about the relationship between a need and a possible recommendation. The user profile should encompass some knowledge structure that supports this inference. An example of such a RS is presented in [8], where the system *Entree* uses some domain knowledge about restaurants, cuisines, and foods to recommend a restaurant to its users. The main advantage using a knowledge-based system is that there is no bootstrapping problem. Because the recommendations are based on prior knowledge, there is no learning time before making good recommendations. However, the main drawback of knowledge-based systems is a need for knowledge acquisition for the specific domain which makes difficult the adaptation in another domain and not easily adapted to the individual user as it is enhanced by predefined recommendations.

2.4 Fundamental Problems of Recommender Systems

Cold Start Problem. The cold-start problem [42] is related to the learning rate curve of a RS. The problem could be analyzed into two different sub-problems:

- **New-User Problem**, i.e., the problem of making recommendations to a new user [32], where almost nothing is known about his/her preferences.
- **New-Item Problem**, i.e., the problem where ratings are required for items that have not been rated by users. Therefore, until the new item is rated by a satisfactory number of users, the RS would not be able to recommend this item. This problem appears mostly in collaborative approaches and could be eliminated with the use of content-based or hybrid approaches where content information is used to infer similarities among items.

  This problem is also related, with the *coverage* of a RS, which is a measure for the domain of items over which the system could produce recommendations. For example, low coverage of the domain means that only a limited space of items is used in the results of the RS and these results usually could be biased by preferences of other users. This is also known as the problem of *over-specialization*. When the system can only recommend items that score highly against a user's profile, the user is limited to being recommended items that are similar to those already rated. This problem, which has also been studied in other domains, is often addressed by introducing some randomness. For example, the use of genetic algorithms has been proposed as a possible solution in the context of information filtering [44].

Novelty Detection—Quality of Recommendations. From those items that a RS recommends to users, there are items that are already known to the users and items that are new (novel) and unknown to them. Therefore, there is a competitiveness between the desire for novelty and the desire for high quality recommendations. One hand, the quality of the recommendations [38] is related to "trust" that users express for the recommendations. This means that a RS should minimize false positive errors and, more specifically, the RS should not recommend items that are not desirable. On the other hand, novelty is related with the "timestamp—age" of items: the older items should be treated as less relevant than the newer ones and this causes increase to the novelty rate. Thus, a high novelty rate will produce poor quality recommendations because the users will not be able to identify most of the items in the list of recommendations.

Sparsity of Ratings. The sparsity problem [1, 22] is related to the unavailability of a large number of rated items for each active user. The number of items that are rated by users is usually a very small subset of those items that are totally available. For example, in *Amazon*, if the active users may have purchased 1 % of the items and the total amount of items is approximately 2 millions of books, this means that there are only 20,000 of books which are rated. Consequently, such sparsity in ratings degrades the accurate selection of the neighbors in the step of neighborhood formation and leads to poor recommendation results.

A number of possible solutions have been proposed to overcome the sparsity problem such as content-based similarities, item-based CF methods, use of demographic data and a number of hybrid approaches [9]. A different approach to deal with this problem is proposed in [40], where the authors utilized *dimension reduction techniques*, such as singular value decomposition, in order to transform the sparse

user-item matrix R into a dense matrix. The SVD is a method for matrix factorization that produces the best lower-rank approximations to the original matrix [29].

Scalability. RS, especially with large electronic sites, have to deal with a constantly growing number of users and items [7, 51]. Therefore, an increasing amount of computational resources is required as the amount of data grows. A recommendation method, that could be efficient when the number of data is limited, could be very time-consuming and scale poorly. Such a method would be unable to generate a satisfactory number of recommendations from a large amount of data. Thus, it is important that the recommendation approach be capable of scaling up in a successful manner [37].

Lack of Transparency Problem. RS are usually black boxes, which means that RS are not able to explain to their users why they recommend those specific items. In content-based approaches [47, 48], this problem could be minimized. However, in collaborative approaches, predictions may be harder to explain than predictions made by content-based models [17].

Gray Sheep User Problem. The majority of users falls into the class of so called "white-sheep", i.e. those who have high correlation with many other users. For these users, it should be easy to find recommendations. In a small or even medium community of users, there are users whose opinions do not consistently agree or disagree with any group of people [11]. There are users whose preferences are atypical (uncommon) and vary significantly from the norm. After neighborhood formation, these users will not have many other users as neighbors. As a result, there will be poor recommendations for them. From a statistical point of view, as the number of users of a system increases, so does the probability of finding other people with similar preferences, which means that better recommendations could be provided [49].

References

1. Adomavicius, G., Tuzhilin, E.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005)
2. Arakawa, K., Odagawa, S., Matsushita, F., Kodama, Y., Shioda, T.: Analysis of listeners' favorite music by music features. In: Proceedings of the International Conference on Consumer Electronics (ICCE), pp. 427–428, IEEE (2006)
3. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. Addison-Wesley, New York (1999)
4. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997). doi:10.1145/245108.245124
5. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence AAAI'98/IAAI'98, pp. 714–720. American Association for Artificial Intelligence, Menlo Park (1998)
6. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adapt. Interact. **10**(2–3), 147–180 (2000). doi:10.1023/A:1026501525781

7. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann (1998)
8. Burke, R.: Knowledge-based recommender systems (2000)
9. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002). doi:10.1023/A:1021240730564
10. Celma, O., Ramrez, M., Herrera, P.: Foafing the music: a music recommendation system based on RSS feeds and user preferences. In: Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR). London (2005)
11. Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of ACM SIGIR Workshop on Recommender Systems (1999)
12. Cox, I.J., Miller, M.L., Omohundro, S.M., Yianilos, P.N.: PicHunter: Bayesian relevance feedback for image retrieval. Int. Conf. Pattern Recognit., **3**, 361 (1996). doi:10.1109/ICPR.1996.546971
13. Doulamis, N.D., Doulamis, A.D., Varvarigou, T.A.: Adaptive algorithms for interactive multimedia. IEEE MultiMed. **10**(4), 38–47 (2003). doi:10.1109/MMUL.2003.1237549
14. Foote, J.: An overview of audio information retrieval. Multimed. Syst. **7**(1), 2–10 (1999)
15. Grimaldi, M., Cunningham, P.: Experimenting with music taste prediction by user profiling. In: Proceedings of Music Information Retrieval 2004 (MIR'04). New York (2004)
16. Herlocker, J.L., Konstan, J.A.: Content-independent task-focused recommendation. IEEE Internet Comput. **5**(6), 40–47 (2001). doi:10.1109/4236.968830
17. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work CSCW'00, pp. 241–250. ACM, New York (2000). doi:10.1145/358916.358995
18. Herlocker, J.L., Konstan, J.A., Riedl, J.: An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Inf. Retr. **5**(4), 287–310 (2002). doi:10.1023/A:1020443909834
19. Hoashi, K., Matsumo, K., Inoue, N.: Personalization of user profiles for content-based music retrieval based on relevance feedback. In: Proceedings of ACM International Conference on Multimedia 2003, pp. 110–119, ACM Press (2003)
20. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management CIKM'01, pp. 247–254. ACM, New York (2001). doi:10.1145/502585.502627
21. Krulwich, B.: Lifestyle finder: intelligent user profiling using large-scale demographic data. AI Mag. **18**(2), 37–45 (1997)
22. Linden, G., Smith, B., York, J.: http://Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003). doi:10.1109/MIC.2003.1167344
23. Liu, D.R., Shih, Y.Y.: Integrating AHP and data mining for product recommendation based on customer lifetime value. Inf. Manag. **42**(3), 387–400 (2005). doi:10.1016/j.im.2004.01.008
24. Logan, B.: Music recommendation from song sets. In: Proceedings of 5th International Conference on Music Information Retrieval, pp. 425–428 (2004)
25. Mandel, M., Poliner, G., Ellis, D.: Support vector machine active learning for music retrieval ACM multimedia systems journal. ACM Multimed. Syst. J. **12**(1), 3–13 (2006)
26. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of Eighteenth National Conference on Artificial intelligence, pp. 187–192. American Association for Artificial Intelligence, Menlo Park (2002)
27. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204. ACM DL'00, New York (2000). doi:10.1145/336597.336662
28. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting web sites. Mach. Learn. **27**(3), 313–331 (1997). doi:10.1023/A:1007369909943
29. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. **13**(5–6), 393–408 (1999). doi:10.1023/A:1006544522159

30. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence UAI'00, pp. 473–480. Morgan Kaufmann Publishers Inc., San Francisco (2000)
31. Popescul, A., Ungar, L.H., Pennock, D.M., Lawrence, S.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence UAI'01, pp. 437–444. Morgan Kaufmann Publishers Inc., San Francisco (2001)
32. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces IUI'02, pp. 127–134. ACM, New York (2002). doi:10.1145/502716.502737
33. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of Computer Supported Collaborative Work Conference, pp. 175–186. ACM Press (1994)
34. Rich, E.: User Modeling via Stereotypes, pp. 329–342. Morgan Kaufmann Publishers Inc., San Francisco (1998)
35. Rui, Y., Huang, T.S., Ortega, M., Mehrotra, S.: Adaptive algorithms for interactive multimedia. IEEE Trans. Circuits Syst. Video Technol. **8**(5), 644–655 (1998)
36. Salton, G. (ed.): Automatic Text Processing. Addison-Wesley, Reading (1989)
37. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of 10th International Conference on World Wide Web, pp. 285–295. ACM, New York (2001). doi:10.1145/371920.372071
38. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: Proceedings of 2nd ACM Conference on Electronic Commerce, pp. 158–167. ACM, New York (2000). doi:10.1145/352871.352887
39. Sarwar, B.M.: Scalability, and distribution in recommender systems. Ph.D. dissertation, University of Minnesota (2001)
40. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system—a case study. In: ACM WebKDD Workshop (2000)
41. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Data Min. Knowl. Discov. **5**(1–2), 115–153 (2001). doi:10.1023/A:1009804230409
42. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR'02, pp. 253–260. ACM, New York (2002). doi:10.1145/564376.564421
43. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 210–217. ACM Press/Addison-Wesley Publishing Co., New York (1995). doi:10.1145/223904.223931
44. Sheth, B., Maes, P.: Evolving agents for personalized information filtering. In: Proceedings of 19th Conference on Artificial Intelligence for Applications, pp. 345–352 (1993)
45. Smyth, B., Cotter, P.: A personalized TV listings service for the digital TV age. Knowl.-Based Syst. **13**, 53–59 (2000)
46. Soboroff, I., Nicholas, C., Nicholas, C.K.: Combining content and collaboration in text filtering. In: Proceedings of the IJCAI99 Workshop on Machine Learning for Information Filtering, pp. 86–91 (1999)
47. Sotiropoulos, D.N., Lampropoulos, A.S., Tsihrintzis, G.A.: MUSIPER: a system for modeling music similarity perception based on objective feature subset selection. User Model. User-Adapt. Interact. **18**(4), 315–348 (2008)
48. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Providing justifications in recommender systems. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**(6), 1262–1272 (2008)
49. Terveen, L., Hill, W.: Human-computer collaboration in recommender systems. In: Carroll, J. (ed.) HCI in the New Millennium. Addison Wesley, New York (2001)

50. Tran, T., Cohen, R.: Hybrid recommender systems for electronic commerce. In: Proceedings of Knowledge-Based Electronic Markets. Papers from the AAAI Workshop, AAAI Technical Report WS-00-04, pp. 78–83. Menlo Park (2000)
51. Ungar, L., Foster, D., Andre, E., Wars, S., Wars, F.S., Wars, D.S., Whispers, J.H.: Clustering methods for collaborative filtering. In: Proceedings of AAAI Workshop on Recommendation Systems. AAAI Press (1998)
52. Vignoli, F., Pauws, S.: A music retrieval system based on user driven similarity and its evaluation. In: Proceedings of 6th International Conference on Music Information Retrieval, pp. 272–279. London (2005)
53. Zhang, Y., Koren, J.: Efficient Bayesian hierarchical user modeling for recommendation system. In: Proceedings of 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 47–54. ACM, New York (2007). doi:10.1145/1277741.1277752

http://www.springer.com/978-3-319-19134-8

Machine Learning Paradigms
Applications in Recommender Systems
Lampropoulos, A.S.; Tsihrintzis, G.A.
2015, XV, 125 p. 32 illus., 6 illus. in color., Hardcover
ISBN: 978-3-319-19134-8