UDC: 004.048

RECOMMENDER SYSTEMS: AN OVERVIEW

Rustamov Alisher
arustamov_88@mail.ru
Bekkamov Fayzi
bfayzi@mail.ru
teacher's of the department of Information security,
Karshi branch of TUIT

Abstract:

Background. *In this article, we look at the key advances in collaborative filtering recommender systems, focusing on the evolution from research focused solely on algorithms to research focused on the broad set of issues surrounding user experience with the recommender. The Internet provides a huge amount of heterogeneous information, and its number is increasing rapidly every year. Related to this is the concept of information overload, when a person is exposed to an excessive amount of information and is not able to effectively process and use information. Recommender systems are trying to eliminate this problem. However, the difficulty of evaluating and selecting relevant information is constantly increasing. Therefore, new recommendation systems using various evaluation strategies are constantly emerging. There are also studies that demonstrate the effectiveness of their use in solving problems with information retrieval and recommendations. Unlike traditional models (such as collaborative filtering), deep learning provides a better understanding of user requirements, item characteristics, and the historical interaction between them. Earlier methods use models such as matrix factorization, SVD (Singular Value decomposition), to recommend them. These systems only work with user ratings. Gradually, recommendation systems were created that also used available information about items.*

The recommender system filters out a piece of information based on user behavior or interest. The recommendation system can predict the interest of the user, as well as predict whether the user will prefer a particular product or not. For both users and service providers, a recommendation system is beneficial as well as effective in increasing the sales of many products. This article explores many of the recommender systems methods.

The large amounts of unfiltered information returned by an internet query calls for filters able to validate and rank the available options. Recommender systems are a software tool designed to qualify the options available and make suggestions that align with the user's requirements and expectations. It also reviews various filtering techniques like collaborative, content based, and hybrid.

Method. *In the article explains the basic approaches of the Recommendation Systems from several perspectives. In terms of input data, output data and various approaches that are used in these systems.*

Result. *Recommender systems are used to estimate user preferences for items that users have not yet seen. Based on the output, we then divide recommendation systems with rating prediction, top-n item prediction, and classification.*

Conclusion. *In this article, the recommended systems have been analyzed both in terms of internal representation and in terms of the methods that these traditional systems use. The rating prediction system seeks to fill in as many missing elements as possible in the matrix containing the ratings that the user has assigned to individual elements in the past. The result of the top-n system is an estimated list of elements of length n. The classification system focuses on classifying candidate items into the correct categories for recommendations.*

Keywords: *recommendation system, natural language processing, collaborative filtering, matrix factorization*.

Introduction. Because users are overwhelmed with information, recommendation systems are a useful and effective information tool that guides you when you discover new products and services, of which there are many. They play an important and decisive role in various information systems to improve decision-making processes [1, 3, 5].

A recommender system can be defined as a program that seeks to recommend the most suitable products (products or services) to individual users (individuals or businesses) by predicting the interests of users in an item based on relevant information about the items, users, and relationships between them.

To understand and analyze the application development of recommender systems, this section briefly describes the main recommender techniques as well as the different types of these systems. Various recommender systems and methods have been introduced since the mid-1990s, and more recently, many recommender systems have been developed for different types of applications. Many researchers and managers recognize that recommender systems offer great opportunities for business, government, education, and others. Proof of this is the fact that these systems are used by large international companies and organizations. Think, for example, LinkedIn, Youtube, Spotify, Amazon and others. There are also many free open source guidelines out there. However, many of these available systems use older techniques such as collaborative filtering.

In general, the output of the recommendation system is a list of recommended items that is generated based on user preferences, item properties, past user interactions with the item, and other information such as temporal and, last but not least, spatial data used for analysis. and predict geographic.

Method. Recommendations models can be divided into several types: collaborative filtering models, content-based recommendation systems, and hybrid recommendation systems.

Collaborative filtering

Collaborative filtering helps people make decisions based on the opinions of others who share similar interests. Collaborative filtering recommends based on the historical interaction of user-element relationships, either explicitly - for example, from previous user ratings (a user-based method), or using implicit feedback

(a subject-based method) - for example, from browsing history. With a custom approach, users are provided with recommendations for products that are popular with similar users. With a substantive approach, the user receives product recommendations similar to those he had in the past (Figure 1) [2, 4].

Similarity between users or products can be calculated using similarity based on Pearson correlation, constrained similarity based on Pearson correlation, cosine-like similarity, or cosine-based measurement. The above methods only calculate users who rated both items when calculating between items. However, this can affect the accuracy of similarity if items that receive a small number of ratings show a high level of similarity with other items. Therefore, an extended version of Domain-Specific Collaborative Filtering was presented, which combines cosine-based measurements with Jaccard metrics as a weighted scheme.

Equation (1) demonstrates the Pearson's matching coefficient:

$$sim(a,b) = \frac{\sum_{p\in P}(r_{a,p}-\widetilde{r_a})(r_{b,p}-\widetilde{r_b})}{\sqrt{\sum_{p\in P}(r_{a,p}-\widetilde{r_a})^2}\sqrt{\sum_{p\in P}(r_{b,p}-\widetilde{r_b})^2}} \quad (1)$$

where the set of users is defined as $P = p_1, \dots p_m$and R as a matrix $n \times m$ assessment $r_{i,j}$ for $i \in 1 \dots m$ и $j \in 1 \dots n$. Symbol (r_x) indicates the average rating of the user x. The equation gives the similarity $sim(a,b)$ for users a and b and returns real numbers in the range $< -1,1 >$. As long as there is a strong relationship between users, the result is close to 1. Conversely, if they have different opinions, the value is close to -1.

Collaborative filtering can be implemented in a memory based form (usually user based) or model. In a memory-based approach, the entire user database is kept in memory and the entire database is scanned in every operation. With this approach, the recommendations are more accurate, but if the system database is too large, this approach is almost impossible, since there are limitations on storing the database in primary memory. In addition, scanning the entire database can take a long time if it is too large.

On the other hand, the model-driven approach does not have the aforementioned memory constraints. In this method, instead of storing the entire database in memory, only a specific set of data is stored in memory, which has already been trained by machine learning methods. While there is a model-based approach in some cases that has similar memory constraints, this approach is more efficient and feasible in the real world.

Thus, the two main areas of collaborative filtering are neighborhood methods and latent factor models. Neighborhood methods focus on calculating relationships between elements, or similarly between users. An item-centric approach estimates a user's preferences for items based on the ratings of neighboring items by the same user. Neighboring products are products that tend to receive the same rating from the same user. Consider, for example, the “Star Wars” book, other science fiction books, and space-themed books can be included in their adjacent items. To get a rating for this book, we look at the surrounding books that the user has rated.

Hidden factor models are an alternative approach that seeks to analyze rankings by characterizing both users and items, most often twenty to one hundred factors, that are derived from ranking templates. For books, putative factors can measure direct measurements such as romance versus science fiction, popular science, or fiction; less well-defined measurements may include character depth or completely unlikely measurements. For each user, each factor measures how much he likes the book, reaching high values for the corresponding book factor. For example, we can consider two hypothetical dimensions: books for men and books for women and books for serious and leisure.

Matrix factorization

As the name suggests, the supposed factorization of a matrix is to find two (or more) matrices from which we get the original matrix if we multiply those matrices by each other. This method is also used in recommendation systems [6].

Matrix factorization is used to find hidden features that emphasize the interaction between two objects, that is, between the user and the subject (book), and one application is to predict the score with collaborative filtering.

When identifying features, we assume that the number of features will be relatively less than the number of users and the number of items. If this were not the case, the user would only be associated with a unique feature, and the recommendation would not make sense, since each of these users would not be interested in items that other users have rated. A similar assumption applies to articles.

Mathematics of matrix factorization

Let us have a set of users U and a set of elements I [7, 8]. Let be R size $|U| \times |I|$ contains all the ratings that users have assigned to items. We also assume that we are looking for K hidden factors. The goal is to find two matrices $P(|U| \times K$ matrix) and $Q(|I| \times K$ matrix), such that their scalar product approximates R:

$$R \approx P \times Q^T = \hat{R} \ (2)$$

Every line P represents the strength of the connection between user and functions. Likewise, every line Q represents the strength of the connection between element and functions. To get a prediction of the item score d_j, user supplied u_i, we calculate the dot product of two vectors corresponding to u_i and d_j:

$$= p_i^T \cdot q_j = \sum_{k=1}^{k} p_{ik} \cdot q_{kj} \ (3)$$

Calculation P and Q begins with the initialization of these matrices with random values on the matrices P_0 and Q_0, calculating how much their dot product differs from the matrix I, then iteratively minimizing the difference. We call this method gradient descent, which aims to find the local minimum of the difference.

The difference is here taken as an error between the estimated estimate and the actual estimate and can be calculated for each user-member pair using the following equation:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{k=1}^{K} p_{ik} \cdot q_{kj})^2 \ (4)$$

We consider squared error because the estimated estimate may be less or greater than the actual one.

To minimize error, we need to know in which direction to adjust the values p_{ik} and q_{kj}. In other words, we need to know the gradient of the actual values, and so we distinguish between two equations that handle two variables separately:

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2\left(r_{ij} - \hat{r}_{ij}\right)(q_{ik}) = -2e_{ij}q_{ij} \text{ (5)}$$

$$\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2\left(r_{ij} - \hat{r}_{ij}\right)(p_{ik}) = -2e_{ij}p_{ij} \text{ (6)}$$

After obtaining the gradient, we can formulate the rules for updating both variables p_{ik} and q_{kj}:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + 2\alpha e_{ij}q_{kj} \text{ (7)}$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + 2\alpha e_{ij}p_{ik} \text{ (8)}$$

Where α - constant, the value of which determines the rate of convergence k minimum. Usually a small value is chosen, for example 0,0002. With a larger step to the minimum, there is a risk of not finding the minimum and fluctuating around it.

Matrix P and Q, where $P \times Q$ approximates the score matrix R, does not give zero ratings for those ratings that are missing, as it might seem. In this factorization, he does not try to find such P and Q, which would accurately reproduce the matrix R. This is just an attempt to minimize the error of the available user-element pair. Accordingly, let T set n- ticks, each of which has the form $(u_i,\ d_j,\ r_{ij})$, then T contains all pairs, the user has an element with a rating belonging to this pair, and the task is to minimize all errors e_{ij} for $(u_i, d_j, r_{ij}) \in T$. In other words, T - this is a training set. The values of other unknown ratings can be determined after you learn how to communicate between users, items, and functions.

Using the above equations to update the values, we can loop through the calculations until the error reaches a minimum. The total error is calculated using the following equation and determines when the process should be completed optimally:

$$E = \sum_{(u_i,d_j,r_{ij})\in T} e_{ij} = \sum_{(u_i,d_j,r_{ij})\in T} (r_{ij} - \sum_{k=1}^{K} (p_{ik}\, q_{kj})^2 \text{ (9)}$$

SVD (Singular Value decomposition)

In the context of singular value decomposition is used as a collaborative filtering algorithm [9]. Singular value decomposition is a matrix factorization technique that is commonly used to reduce the number of properties of a dataset by decreasing the dimension with D before K, where $K < D$. Matrix factorization is performed on the evaluation matrix $R = U \times I$, where U - number of users, but I - the number of elements.

Each vector can be represented by a vector q_i. Similarly, each user can be represented by a vector p_u, and so their dot product is the expected estimate:

$$hodnoceni = \hat{r}_{ui} = q_i^T \cdot p_u \ (10)$$

$$minimum(p, q) = \sum_{(u,i)\in K}(r_{ui} - q_i^t \cdot p_u)^2 \ (11)$$

To better generalize and prevent overtraining of the training sample, this method calculates the control coefficient λ, which is calculated as the product of the squares of the sum of the vectors of users and elements and is a penalty function. Then the equation for the minimum will look like this:

$$minimum(p, q) = \sum_{(u,i)\in K}(r_{ui} - q_i^T \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \ (12)$$

To illustrate this factor, we can imagine an edge case where we only have a low user rating assigned to an item and we have no other user ratings. The algorithm then minimizes the error by assigning a large value to the vector. As a result, all ratings of this user for other films will be low. However, this is undesirable. Adding the size of the vectors to the equation minimizes setting large values for the vectors and thus ensures that such situations are avoided.

The algorithm uses some characteristics of the dataset to reduce the error between the predicted and actual rating values. Specifically for each pair of custom elements (u, i) we can get 3 parameters. Parameter μ as the average rating of all elements, b_i as average item rating i, from which the value is subtracted μ and b_u, which means the average rating given by the user u, from which the value μ is also deducted. Then the equation for the expected estimate will look like this:

$$\hat{r}_{ui} = q_i^T \cdot p_u + \mu + b_i + b_u \ (13)$$

The resulting minimization equation will have the following form:

$$minimum(p, q, b_i, b_u) = \sum_{(u,i)\in K}(r_{ui} - q_i^T \cdot p_u - \mu - b_i - b_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2) \ (14)$$

Content-based recommendation systems

A content-based recommender system is usually based on comparing individual elements and useful information about users. Different types of information can be taken into account, such as images, videos or texts (user reviews, texts of books or music, and others) (Figure 1).

Systems implementing content-based recommendations analyze a set of documents and descriptions of items previously assessed by the user and create a model or profile of user interests based on the properties of objects assessed by the user. A profile is a structured form of user interests that is used to recommend new interesting things to users. The recommendation process mainly consists of adapting user profile attributes and content object attributes. The result is a relevance check that shows the user's level of interest in the object [10, 11].

It is very important for the efficiency of the information access process that the profile accurately reflects the user's preferences. For example, they can be used to filter search results when the system decides whether a user is interested in a particular website. Otherwise, this page will not be displayed.

The recommendation process is carried out in three stages, each of which is processed by a separate subsystem:

- content analyzer - if we work with unstructured information (for example, with texts), it is necessary to perform a certain type of preprocessing in order to get structured relevant information. The main task of this component is to present the content of elements (documents, websites, reports, product descriptions, reviews) from information sources in a convenient form for further processing steps. Data items are analyzed with feature extraction techniques to move information from source space to destination (eg, web pages are then represented as key vectors). This view then serves as input for the profile learner and the filter part.
- profile learner - this module collects data representing user preferences and tries to summarize the data to create a user profile. The generalization part is mainly in charge of machine learning methods that can derive a user interest model based on elements that users have liked or disliked in the past. For example, a student of a profile who recommends a website might implement an appropriate feedback method that combines a vector of positive and negative examples into a prototype vector representing the user's profile. An example of training data is a website that a user has left positive or negative feedback about.
- filtering section - this module uses the user profile to design the relevant items that it gets by comparing the profile view with items to recommend. The result is a binary or continuous score, calculated using some similarity metrics, from which a list of potentially interesting items is generated. In the above example with web pages, the comparison is done by calculating the cosine similarity between the prototype vector and the element vector.

To create or update the profile of the active user u_a (the user to whom the recommendation should be given), his replies to items are somehow collected and stored in the feedback repository.

These reactions, called annotations or feedback, are used in conjunction with descriptions of related items during model training, which is useful for predicting the value of newly introduced items. Users can also explicitly define their areas of interest when initially creating a profile or account without feedback.

You can use two different methods to record user feedback. When the system requires the user to explicitly rate items, we usually refer to this method as explicit feedback. The second method, called implicit feedback, does not require active user involvement in this regard. Feedback arises as a result of monitoring and analyzing user actions. An explicit rating indicates how important or interesting an item is to a given user [12].

There are three main approaches to getting explicit feedback:

- like / dislike - Items are classified as “relevant” or “irrelevant” using a simple binary rating scale.

•ratings - the rating of an item is presented here according to a discrete numeric scale (for example, from 1 to 10, the more the more relevant the item is to the user), or you can use a symbolic rating, which is then displayed on a numerical scale, or you can use a symbolic rating, which then displayed on a numeric scale.

•text rating - comments on individual items are collected and presented to users as a tool to facilitate decision-making. For example, customer reviews on Amazon.com or eBay.com can help users decide if an item has been awarded by the community. Text comments are useful, but they can overwhelm the active user as they must read and interpret each note to decide if it is positive or negative, and to what extent.

Explicit feedback has the advantage of simplicity, although the use of numeric / character scales increases the cognitive load on the user and may not be sufficient to capture the user's opinion of objects.

Implicit feedback methods are based on assigning a relevance score to specific user actions associated with an item, such as saving, deleting, printing, bookmarking, and others. The main advantage is that they do not require direct user interaction, although rejection may occur.

Creating a user profile

To create an active user profile u_a, it is necessary to define the training set $T\ R_a$ for the user u_a. $T\ R_a$ is a set of pairs of type $< I_k, r_k >$, where r_k is the rating assigned by the user u_a representing the element I_k. After presenting a set containing a view of rated items, the profile student uses a supervised learning algorithm and generates a predictive model - a user profile - that is typically stored in the profile repository for later use by the filtering part. After a new item is submitted, the filter section predicts whether the item is in the active user's area of interest by comparing properties in the new item's view with those stored in the user's profile. So part of the filtering involves some strategies for sorting potentially interesting items according to relevance to the user's profile. The items with the highest rating are included in the L_a recommendation list, which is provided to the active user u_a. As user preferences change over time, updated information must be translated into the learner's profile part in order to automatically update the user's profile.

Further feedback is collected after providing users with specific recommendations, when users have the opportunity to express their satisfaction or dissatisfaction with the items in the L_a list. After collecting this information, the learning process is performed again on a new training set, and then the user profile is adapted to the changes in the interests of the user. The feedback learning method allows you to respond to the dynamic nature of user preferences.

Submission of items

Items that should be recommended to users can be represented by a set of properties, attributes, or functions. For example, book recommendations use the following attributes to describe books: author, genre, title, original title, year of publication). Each element is defined by the same set of attributes and the set of values that the attributes can take. Then we get the element represented by structured data. In this case, we can use a variety of machine learning techniques to study the user's profile.

In most content-based filtering systems, item descriptions are text properties extracted from web pages, emails, news articles, or product descriptions. Unlike structured data, there are no fully-valued attributes. Text functions create a number of difficulties when learning a user profile due to the ambiguity of natural language. The problem is that traditional keyword-based profiles fail to capture the semantics of user interests because they are mostly based on string comparisons. If a string or any morphological variation is found in both the profile and the document, a match is made and the document is considered relevant. However, string comparisons suffer from these problems. Polysemy (the presence of several meanings in one word) and synonyms (several words with the same meaning).

As a result, due to synonyms, important information may be omitted if the profile does not contain the exact keywords contained in the document. And because of polysemy, bad documents can be considered relevant. Semantic analysis and its integration into personalization models is one of the most innovative and interesting approaches proposed in the literature to address these problems. The key idea is to adopt knowledge bases such as lexicons, ontologies, for annotating elements and presenting profiles, in order to obtain a "semantic" interpretation of the information users' needs.

Hybrid recommender systems

To achieve better performance and overcome the shortcomings of traditional recommendations, hybrid recommendations have been proposed that combine the best characteristics of two or more recommendation techniques into a single hybrid technique. The most common approach in existing hybrid system methods is to combine the collaborative filtering recommendation method with other recommendation methods to avoid cold start, sparse data, or data scalability issues. These systems use both user ratings information and available information and data about users and products to generate the best recommendations. Both implicit and explicit information (obtained when creating a user profile) are taken into account (Figure).

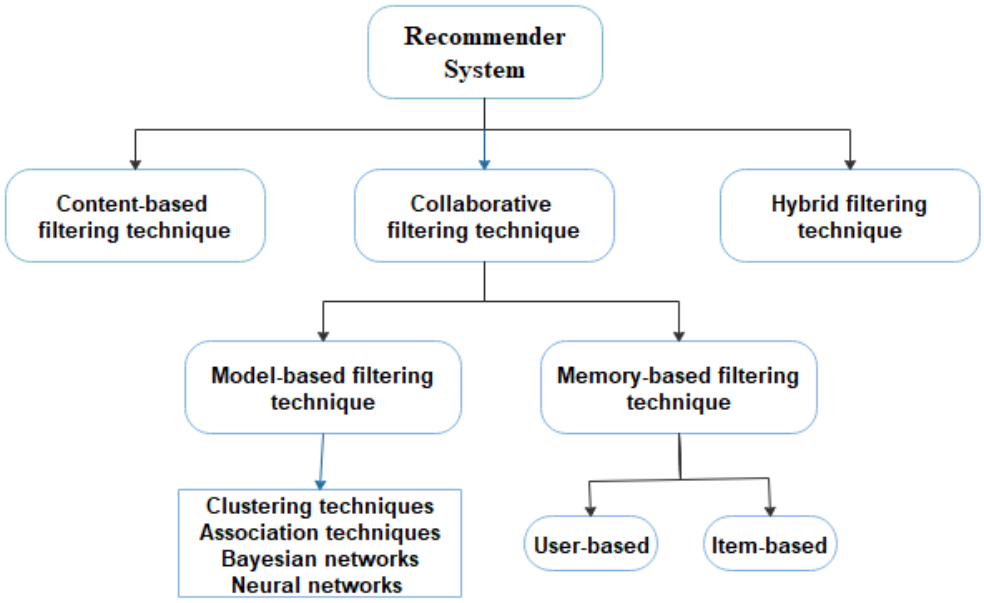

Figure. Recommendation techniques

Result. The main problem for recommender systems is the so-called cold start problem. In order for the system to adapt to the user, he must know what the user wants and what is important to him. This is necessary for content-based filtering to make decisions about items similar to those that users have liked in the past. What if we still don't know anything about users who have just started using the system? The designers of these systems tend to solve the problem by either asking users to rate the elements first, or by offering them a demographic questionnaire from which certain stereotypes can be inferred (for example, older people listen to classical music more). Therefore, we require users to explicitly fill out a user profile.

Both methods require user effort. It is also not easy to decide which elements should be rated by users. Also, stereotypes can be relatively bad and offensive (for example, some people prefer popular music and don't want to be considered older). We will gradually learn about the taste of the new user, for example by evaluating our recommended items or using a more implicit method, checking if they are wasting time on these items or not. We make recommendations to the new user so that the group of existing users can be satisfied, including the new user (or more precisely, the person we now consider to be a new user). The weight assigned to the new user will be low at first because we don't know much yet, and will gradually increase.

The data scarcity problem is caused by the lack of transaction data and feedback, which limits the usability and success of collaborative filtering and other methods. This problem can be minimized, for example, by direct or indirect similarity between users and by calculating a similarity matrix based on the relative distance between user ratings. Recently, however, new recommender systems have been created that try to minimize this problem as much as possible. To do this, they use machine learning and complement the disadvantages of collaborative filtering with low density user data.

Conclusion. Information overload is a major problem in information retrieval systems. The recommender system solves the problem of information overload and opens up new possibilities for obtaining personalized information on the Internet and for users' access to products and services. This paper describes content-oriented and collaborative filtering, and also compares their strengths and weaknesses with hybrid filtering to improve the efficiency of the system. We also discussed various algorithms for generating recommendations and measuring the quality and performance of the recommendation system.

REFERENCES

1. Bohnert F., Schmidt D.F., Zukerman I. Spatial Processes for Recommender Systems. [Online; navštíveno 29.12.2017].

2. Lu J., Wu D., Mao M. aj.: Recommender System Application Development s: A Survey 2019

3. Sarwar B., Karypis G., Konstan J. aj.: Item - based collaborative filtering recommendation algorithms. 2018

4. Resnick P., Iacovou N., Suchak M. aj.: GroupLens: an open architecture for collaborative filtering of netnews. 2018

5. Shambour Q., Lu J. hybrid trust - enhanced collaborative filtering recommendation approach for personalized government - to - business e - services, International Journal of Intelligent Systems. 2017

6. Madadipouya K., Chelliah S. A Literature Review on Recommender Systems Algorithms, Techniques and Evaluations. BRAIN: Broad Research in Artificial Intelligence and Neuroscience, ročník 8, č. 2, July 2017.

7. Lops P., de Gemmis M., Semeraro *G. Content-based Recommender Systems: State of the Art and Trends. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA, 2011, ISBN 978-0-387-85819-7.*

8. Lisa Wenige, Johannes Ruhland, Retrieval by recommendation: using LOD technologies to improve digital library search, © Springer Verlag GmbH Germany 2017

9. Mingdan Si, Qingshan Li, Shilling attacks against collaborative recommender systems: a review, Artificial Intelligence Review (2020) 53:291–319

10. Hui Li, Yan Gu, Saroj Koul, A Review of Digital Library Book Recommendation Models, 2015

11. Joseph A. Konstan, John Riedl, Recommender systems: from algorithms to user experience, © Springer Science+Business Media B.V. 2012

12. Emmanouil Vozalis, Konstantinos G. Margaritis, Analysis of Recommended Systems Algorithms, 2014