Chapter 5
A Survey on the Scalability of Recommender Systems for Social Networks

C. Sardianos, N. Tsirakis and I. Varlamis

Abstract It is typical among online social networks' users to share their status, activity and other information with fellow users, to interact on the information shared by others and to express their trust or interest for each other. The result is a rich information repository which can be used to improve the user experience and increase their engagement if handled properly. In order to create a personalized user experience in social networks, we need data management solutions that scale well on the huge amounts of information generated on a daily basis. The social information of an online social network can be useful both for improving content personalization but also for allowing existing algorithms to scale to huge datasets. All current real-world large-scale recommender systems have invested on scalable distributed database systems for data storage and parallel and distributed algorithms for finding recommendations. This chapter, focuses on collaborative filtering algorithms for recommender systems, briefly explains how they work and what their limitations are.

Keywords Collaborative filtering · Recommender systems · Scalability
High-performance

5.1 Introduction

A social network is a platform that facilitates individuals, connected through social relations, such as family, friends, and colleagues [1, 2] to communicate with each

C. Sardianos (✉) · I. Varlamis
Department of Informatics and Telematics, Harokopio University of Athens,
Omirou 9, Athens, Greece
e-mail: sardianos@hua.gr

I. Varlamis
e-mail: varlamis@hua.gr

N. Tsirakis
Department of Computer Engineering & Informatics, University of
Patras, Rio, Patras 26500, Greece
e-mail: tsirakis@ceid.upatras.gr

N. Dey et al. (eds.), *Social Networks Science: Design, Implementation, Security, and Challenges*, https://doi.org/10.1007/978-3-319-90059-9_5

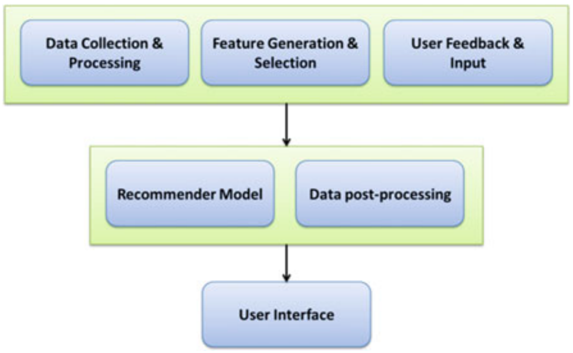

Fig. 5.1 Components of the Recommender Systems

other. When social networks expand at world-scale, as it is the case with the most popular social networking sites and applications (e.g. Facebook, YouTube, Instagram, Twitter etc.), the supporting software and hardware must scale too, in order to support the millions of users. According to [3], scalable computing systems maintain consistent performance under an expanded workload. Referring to scalability in social networks, we expect no compromise on performance from a social network and also expect a standard level of latency, without significant changes, no matter how its provision changes as new users are connecting or new content is being added.

Undoubtedly, the continued and differentiated expansion of social networks has changed the way in which users communicate and interact with each other. There are now more ways to exchange ideas, share opinions and learn in collaboration. In the same time, the need for new features in social networking applications, which will allow user to follow this expansion and delve with the increasing information, has emerged. Trying to meet this need, Recommender Systems have come to provide a solution and there are many techniques that have been used for different approaches to this problem. Such systems become integral parts of most social networking applications, takes advantage of the stated or implicit preferences of users, the additional information concerning their interaction with other users and the content they access or share, and makes suggestions for new content, contacts or actions. They usually act as systems trying to predict the user's score for any potential item [4] or the user's opinion based on some item aspects.

As depicted in Fig. 5.1, the main components of a recommender system are:

- Data Collection and Processing: This is the task for the collection of data which most of the times are large.
- Feature Generation and Selection: In this task algorithms are responsible for feature generation and selection that can be performed either pre-calculate these features or dynamically generates them.
- User Feedback and Input: In this task the user is invited through the system interface to provide ratings for items in order to construct and improve his model.
- Recommender Model: This is the main part of any recommender that orchestrates the recommendation algorithm with all the previous data that it gets.

- Data post-processing: Data post-processing is used in many systems in order to optimize any measure generated by the model.
- User Interface: This is the final step of any system and is the component where the user interacts with the system.

Lately, the field of application of recommender systems has expanded in a wide area of domains such as creating movie or music recommendations, suggesting related news, finding useful scientific research papers, recommending possible social connections or potential products users could be interested in buying. But the type of domains recommender systems are used for are not limited to the above. There have been developed many domain-specific RSs such as for finding experts based on a query string and the domain characteristics [5], or potential researcher for collaborating with [6], even for supporting suggestions on loans etc. [7], or just simply suggesting pages of interest in Twitter [8]. In general, RSs aim to solve the information overload problem for the users of a social network by recommending items, which can either be content, products, services or even other users. In general, the input of the process for creating recommendations is a set of users U and a set of items I, whilst the function f that represents their relation is:

$$f : U \times I \to R \tag{5.1}$$

This means that the recommendation algorithm "rates" each user-to-item relation with a predicted score that represents the interest of user u for item i, which in terms is the rating that the system predicts the user would give to this item and is created in one of the following ways: (i) by defining a similarity between the user profile and the item, using their content information (e.g. description, stated preferences etc.), (ii) by defining a similarity between user profiles and then using only users with similar profiles to the target user to predict item ratings, (iii) by filtering out items of low interest to the target user [9]. In the latter case, information filtering methods are based on user demographic information (demographic), the content of previously rated items (content-based filtering [10]) or simply the ratings and the ratings provided by other users, also referred as collaborative filtering [11] and are able to predict unknown item-ratings for a user. Several combinations of more than one technique from the aforementioned categories have also been proposed in the literature resulting to hybrid recommender systems [12]. Based on the predicted ratings for the unseen items, recommender systems create a set of items to be recommended to user.

The success of Recommender Systems is undoubtedly over the last ten years or more, but any current real-world large-scale recommender has to consider two of the main points of concern: The quality of the produced recommendations, which is defined by the actual recommendation accuracy, and the scalability, which is limited by the computational cost to provide these recommendations [13]. Scalability is a problem associated with recommendation algorithms (more often in collaborative filtering techniques) because computation normally grows at a linear rate to users and items [14]. A recommendation technique that can be efficient for a limited number of users and items may fail to produce satisfactory recommendations when the volume of the data increases. In general, some of the most important characteristics of collaborative filtering are the ability to generate personalized recommendations based on the user's prior activity in the system, or the activity data of other users that seem to have similar tastes to the given user and the ease of incorporation depending on the selected approach. Collaborative filtering algorithms can be categorized into three major types based on its functionality (Memory-based, Model-based and Hybrid), that are fully analyzed in Sect. 5.2. While many recommender system algorithms have been developed, unless the system can reach unusually high diversity, each algorithm can face different types of problems. One of the main issues collaborative filtering algorithms face these days is scalability and data management due to the exponential growth of user data all over the web.

While product ratings can be found in many product review sites, another useful information provided by social media is the social network information, user-level interactions (likes/dislikes, comments etc.), that is a useful type of information for improving RS predictions. This is based on the idea that a user's circle of friends directly affects the user's ratings. When using this social information, we can use friendship information alone, combine it with user provided ratings, or even filter recommendations using social information. In the process of dealing with scalability and time performance issues that recommender systems usually face, using user profile information for computing user similarity and applying user clustering can be a useful approach. Other approaches build on dimensionality reduction techniques in order to reduce the problem complexity (e.g. Singular Value Decomposition-SVD). When social and other information is added to item ratings the complexity of the recommendation problem increases and scalability solutions have to be reconsidered.

This chapter focuses on collaborative filtering algorithms for recommender systems and surveys the various alternatives for processing huge and sparse rating graphs, with or without external knowledge. The section that follows briefly illustrates the steps of the recommendation process and briefly describes the most widely adopted techniques. Section 5.3 lists the open challenges for recommender systems. Section 5.4 targets on the solutions that have been proposed in the literature for handling recommendations over large data and more specifically in algorithms and implementations that scale up to millions of users and items and billions of ratings. The solutions fall into three main categories: i) those that focus on Factorization of the user-to-item ratings' Matrix and propose parallel and distributed implementation, ii) those that partition the huge set of ratings into clusters taking advantage of external knowledge from the social network (e.g. user or item content similarity, social network bonds etc.) and thus split the recommendation problem into smaller parts, iii) hybrid methods that build mainly on external knowledge trying to boost the results of the distributed Matrix Factorization methods. Finally, Sect. 5.5 summarizes the chapter and highlights the future challenges in the field of recommender systems for social networks.

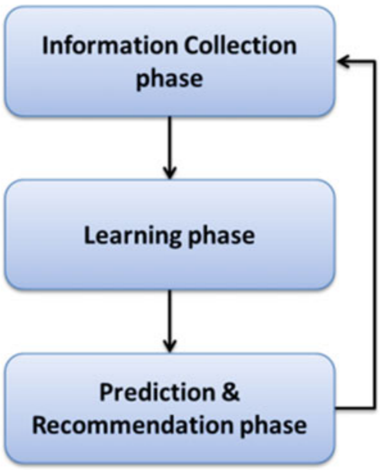
Information Collection phase
Learning phase
Prediction & Recommendation phase

Fig. 5.2 Phases of recommendation process

5.2 Recommendation Techniques

From a system-oriented perspective, the recommendation process takes as input a collection of users and their ratings for a collection of items and produces a per-user personalized set of recommended items. In order to achieve this, a recommender system operates in three stages, which is depicted in Fig. 5.2 and detailed in the following.

Information collection phase: In the first phase, the system collects any relevant information about the users in order to create their profile. Depending on the type of the system, this information must have a minimum size and detail so as the desired model that will be used in the recommendation phase could be prepare. There are systems with specific information collection protocols under which they use this information in order to determine the current state of knowledge and support any decision making for learning any available user profile information for all users taking part in the recommendation process. The information used as input by Recommender Systems in order to prepare a full picture of their users. Such types are either high quality explicit feedback, which includes explicit input by users regarding their interest in items or implicit feedback by inferring user preferences indirectly through observing user's behavior [15].

Learning phase: In the second phase, the system feed the collected information to a learning algorithm that filters and exploits users' features/attributes that will better serve in the recommendation phase. In other words, the system builds the model which is actually an abstraction of the relationship between the items and users.

Prediction and Recommendation phase: This last phase predicts and/or recommends what kind of items the user may prefer. This can be performed either directly, based on the dataset collected in the first phase of information collection, which results to the memory-based or model-based approaches or combined with data that refers to other user activities and preferences, which results to hybrid approaches [12], which are all explained in the following section. The user reaction to the recommended items is continuously recorded and used as a feedback that improves the recommender system performance over time.

In the case of Recommender Systems for Social Networks, the information provided within the social network serves for three main objectives: (a) improving the quality of predictions and recommendations [16, 17], (b) proposing or generating novel Recommender Systems [18, 19], and (c) elucidating the most significant relationships between social information and collaborative processes [20, 21].

There are three main kinds of recommendation techniques based on the information employed for filtering out items of low interest to the target user, predicting item ratings and producing recommendations [9, 22]: (i) Collaborative Filtering techniques that mainly focus on the user-item ratings, (ii) Content-based Filtering techniques that employ additional content for users and items and define several similarity measures and matching models to produce recommendations and (iii) Hybrid techniques that combine the merits of both worlds.

5.2.1 Content-Based Filtering

Content-based filtering is also referred to as cognitive filtering. In content-based filtering techniques, an item is recommended to a user when similarity between this item and the items user had already expressed his preference in the past is high. More specifically, the recommendation is being made by comparing the items' content description. The content of each item is represented as a set of descriptors or keywords and the user profile is represented with the same keywords and is created by examining the content of items which have been seen by the user in the past. In order to use content-based filtering, the similarities for all items are being computed, the items get ranked based on these similarities and the top-N ranked items are finally get recommended to the target user.

5.2.2 Collaborative Filtering

Collaborative filtering (CF) is considered as one of the leading approaches for creating recommendations and that is why it is used by some of the largest commercial platforms. The recognition of its usability is shown by the fact that many variations and techniques have been developed for this purpose. The basic idea behind this technique is that a user provides his preferences in the form of ratings for the available items, either implicitly or explicitly and a customer, who seemed to have similar preferences in the past, would probably still have the same preferences. A basic advantage of this kind of techniques is that there is a limited or no need for semantic information in order to produce recommendations. This is the reason behind their popularity among major social network-based applications including Amazon, Netflix, iTunes, IMDB etc.

Collaborative Filtering (CF) is based on user (or item) similarity [23] measured over the users' (or items') rating information. The aim of CF systems [22] is to predict users' interest for items they have not yet reviewed, based on the ratings they have provided for other items and the ratings of other users for all items. The items with the highest predicted ratings are recommended to the user.

Collaborative Filtering systems can be categorized in two major categories: the memory-based that search for the top-k similar users (i.e. users with similar ratings on the commonly evaluated items) or items (i.e. items that have been rated similar to the items preferred by the user) and use only this information to create their recommendations and the model-based, which use all ratings to train a model for predicting user's preferences for an unseen item.

5.2.3 Hybrid Techniques

Hybrid techniques combine more than one filtering strategies, which are implemented as sub-components of the recommender system. This kind of techniques try to integrate characteristics from collaborative filtering and content based techniques with an ultimate goal of enhancing recommendation quality and overcome the disadvantages of the single filtering techniques. Hybrid filtering is classified in six categories (i) mixed hybrid (ii) weighted hybrid (iii) switching hybrid (iv) cascaded hybrid (v) feature-combination hybrid and (vi) meta-level hybrid. One simple approach, proposed by [24] is to create two sets of ranked recommendations (one with each technique) and then combine them to produce one final list.

The success of deep learning networks in several application domains, also opened a new research field for hybrid recommender systems, which feed the deep learning networks both with rating and content information about users and items and improve the quality of recommendations [25]. Although they better solve many several know issues recommender systems face, such as the cold-start problem [26, 27] or the ratings matrix sparsity [28], they still do not handle scalability issues [29].

5.3 The Challenges of Recommender Systems

Some of the most challenging issues Recommender systems face are scalability, diversity and the long tail, sparsity and of course the cold-start problem. In addition, content-based filtering faces the problem of content analysis in the cases where the sets of items lack of a respectively sufficient set of features. This is justified by the fact that the accuracy of the content-based filtering predictions depends on the amount of information used for classifying the items. Finally, the combination of collaborative filtering and content-based filtering approaches in an efficient way still remains an open issue in many hybrid filtering cases. Some major of the already mentioned challenges are described in more details in the next subsections.

5.3.1 Cold-Start Problem

Cold-start problem is one of the major issues for techniques like collaborative filtering and content-based filtering [30]. During the learning stage of these methods, the analysis of information derived by the user's profile at the system, so in scenarios where this information is provided directly by the user, the existence of users with little to none info available is very common. Specifically in collaborative filtering, the problem occurs when the system tries to predict the score of an item while this item may not have been rated at all by any user. Of course the same issue occurs when new items (and thus never rated before) are being added into the system, which is also known as first-rater problem.

When a new user comes into the system, is very difficult for the recommender system to produce prediction, since the user preferences is yet unknown and there are no items in his history, so both collaborative and content-based filtering can hardly compute any user similarity or create any content profile information respectively. This kind of problems can be eliminated by using hybrid approaches, which are presented in the techniques section, since each technique can be more tolerant to this issue.

5.3.2 Data Sparsity

According to [11], some of the largest commercial sites use recommender systems in order to provide recommendations over very large sets of products. When input data (i.e. user-to-item ratings) are not sufficient for calculating user (or item) similarities then we face the data sparsity issue. This constitutes a critical point in the process of creating high accuracy recommendations which may consequently limit the overall applicability of such systems especially in applications with large and sparse data sources. In other words, the items that have not yet received any ratings from a sufficient number of users, in the case of newly added items for example, can not contribute to the recommendation process as they can not be recommended to any user.

This issue occurs as a result of lack of enough information. Representing the entities that participate in the recommendation process (users and items) as a matrix, it is clear that when the data face the sparsity issue, any user vector that contains the user's ratings for all of the items, will contain more zeros. That means that looking for the positions only of the rated items will produce a more compact format. This in terms leads to reduced demand for memory processing resources.

5.3.3 Synonymy

Word synonymy and polysemy are two major issues for many text mining algorithms that depend on text similarity [31]. Consequently it affects content-based recommender systems that rely on textual descriptions of items or user profiles in order to find similarities. For example, a content-based recommender system is unable to understand that the terms "scary movie" and "horror film" in the description of two items have similar meaning, and fails to associate the two items.

The ambiguity introduced by synonymous and polysemous terms in descriptions decreases the recommendation performance and must be considered. Latent semantic analysis techniques [11] and knowledge-based methods are used to overcome the synonymy problem and improve the content-based recommender systems' performance.

5.3.4 Attacks

The recommender system may be vulnerable to attacks that make it to recommend items that otherwise wouldn't be recommended. There are cases where users have the ability to give recommendations on their own. This means that they can give positive recommendations for their items or friends and also negative for competitor's items or friends. This normally shouldn't be allowed. These are called shilling attacks [32].

There are some other attacks, where users exploit the recommendation mechanism by creating fake profiles and providing false ratings to items in order to promote items of their interest. These are called push attacks. Finally, there are attacks that are called nuke attacks which try to make the recommendation algorithm nonfunctional and thus, stop the recommendation process. Due to these kinds of threats, any production level recommender system should be accompanied by some kind of security mechanism against these attacks [33].

5.3.5 Privacy

Privacy plays an important role regardless of the system type. The concern on keeping data private is to identify the limits between privacy and data access for creating personalized modern applications. Of course, defining these limits is a difficult task, as more data is needed for creating a more personalized profile. This is why these limits are easily violated or ignored in recommendation techniques [34], while profile data are gathered and processed. So privacy protection is a crucial point at the process of the data by every recommender system for personalized systems. In order to provide accurate and valuable recommendations, most techniques require using as much information as possible about the user. Of course there are approaches that try to solve this problem. Randomized transformations of the user-item rating matrix, preserve users' differential privacy [35] without significantly degrading the performance of collaborative filtering algorithms.

5.3.6 Explanation

Explanation, or interpretability of recommendations, is also a characteristic of recommender systems. An instinctive argumentation like "since you already like these movies, you will also like this one as well" gets easily understood by the users, no matter the precision of this statement [36]. Unfortunately, there are systems don't provide good explanations and sometimes are not inspiring users to trust the recommendations and increase their satisfaction. Explanations need to be considered as a must in recommendation systems in order to help the user understand how the system works and take part in the recommendations that are made by giving feedback where is needed. An easy way to provide good explanations is through A/B testing. Satisfaction can also be measured indirectly, measuring user loyalty [37].

5.3.7 Stability

The stability problem occurs when the extent of changes in the recommendation algorithm predictions grows. A stable recommender can make users trust more the system as they will get consistent predictions, whereas a recommender that provides predictions that change over time can confuse users. Stability can be measured if we compare the predictions between two periods during which new ratings have added. Several attempts discount the weight of older ratings by time, in order to reduce their influence to the final recommendation [38] but again they do so at the risk of losing information about permanent, long-term interests that are sporadically expressed.

5.3.8 Scalability

Scalability issues starts to exist when the total number of users and items in the system grows excessively, above the level that traditional Collaborative Filtering algorithms reach their limits of performance. This is the point where the computational resources needed for processing become extremely high to be practically used [11]. The problem can be solved with dimensionality reduction techniques, like SVD, that apply several matrix factorization steps in order to produce good quality recommendations with fewer resources, achieving high quality results with better scalability.

When additional information exists, such as user profile, social network information etc., data partitioning can be applied to the original user-item rating matrix and parallel and distributed algorithms can be used to allow maximum scalability. Nayebzadeh et al. proposed an improved version of Collaborative Filtering [39] to deal with the disadvantage of cold-start problem. Their approach proposes the creation of a model that combines the user preferences as expressed for example in the user profiles, the item acceptance which describes the estimation for a given item *i* that will be rated with value *k* by the users, and the friend inference which describes the relation of any given target user with his neighbors/friends based though on the commonly rated items, and based on this probabilistic model try to predict the user's' ratings for the given items. Although the results of the experiments suggest that this model can perform better or equal to the traditional CF, since the evaluation was performed over a heuristic dataset, this needs to be tested to actual datasets of real large-scale networks to verify the level of solution to the scalability problem.

More details about scalability and possible solutions are presented in the next section.

5.4 Scalability of Recommender Systems for Social Networks

The scalability of an information system is subject to the available resources, the architecture it implements and the algorithms it employs. Similarly, the scalability of a recommender system for social networks relies both on the resources that are available and the system architecture that may allow for distributed or parallel data management and processing, but also on the algorithms that are employed and whether they can adapt to the scalable architecture or to large data sizes. In the former case, parallel and distributed algorithms that must be designed, whereas in the latter case, new algorithms that can achieve comparable performance without using all the available data must be created.

Multimedia content is the second dimension input of recommendation engines over the web after the user dimension. Especially after the prevalence of smartphones that provide multiple ways of accessibility to different content, social media have taken advantage of it and are continually try to provide recommendation based on this different type of data. The challenges though are many. Multimedia data often contains a lot of high-level semantic meanings. Moreover, the scale of data is big with high growth rate, and this requires huge amount of computing resources. Finally, when we referring to multimedia then we are talking about different kinds of content like image, video, audio, text or combinations of them. The majority of techniques in this area exploit high level metadata which is extracted from low level features either in an automatic or a semi-automatic way. This data is compared with user preferences at the end.

5.4.1 Scalability and Data Management

Database Management Systems (DBMS) have a crucial role in social networking applications, where a huge amount of data, produced and consumed at multiple, geographically distinct locations, must be stored, retrieved and delivered efficiently. Centralized Relational DBMSs are designed to provide guaranteed consistency but can hardly scale-up to the requirements of worldwide social networks. These limitations are the reason that popular social networks such as Facebook, and Twitter have explored alternative Distributed and NoSQL data storage systems [40] such as Cassandra [41], Megastore, PNUTS, Dynamo, MongoDB, COPS and SCADS [42].

Although RDBMSs are "not cloud friendly" due to the relationships and dependencies among stored data, NoSQL databases can scale better even when running on commodity hardware with increased fault-tolerance when using cloud infrastructure. However, they also face several challenges that relate to data partitioning and replication [43, 44] and middleware solutions have been developed for this purpose. For example, SPAR [45], a social partitioning and replication middle-ware, which has been designed for supporting social networking applications. SPAR supports different data partition methods (both random and social graph based) and data-stores (both Key-Value store - Cassandra and a relational database -MySQL) and allows scalability, without restricting to a specific database implementation.

5.4.1.1 Data Partitioning

As addressed in [13], the problem of applying Collaborative Filtering to large product-rating graphs is still an open and challenging issue. Collaborative Filtering methods generate recommendations based on product-review information and ratings history for users and products. In the core of their functionality, they need to process a matrix, also represented as bipartite graph, which stores the user ratings for the products. In practice, however these methods strangle to scale to large bipartite graphs. In the following, we assume that both representations -matrix and bipartite graph- are equivalent.

Over the past years, there has been a bulk of research interest regarding several Collaborative Filtering techniques and data partitioning methods that allow existing algorithms to scale up to a larger scale. The technique proposed in [13] predicts the performance of Collaborative Filtering algorithms using the structural features of the bipartite graphs and a machine learning approach in order to find the best partitioning of the original graph. To this end, the authors employ graph partitioning in order to split the original bipartite product-rating graph to smaller sub-graphs and apply the Collaborative Filtering techniques in a parallel setup. With the use of Lenskit Recommender library and Apache Spark [46], they are able to scale the CF-based recommender to large-scale bipartite ratings graphs. In the partitioning step, they take advantage of the undirected social graph, which contains friendship relations between users in the social network. This graph is partitioned first and user partitions are created. Then the bipartite graph is split into sub-graphs by keeping in each bipartite partition only the ratings provided by the users of the respective user partition. In order to deliver good recommendations, the authors generate different numbers of user partitions (and rating sub-graphs respectively) and then predict CF performance in order to determine the best partitioning scheme. For CF performance prediction, they consider additional (bipartite) graph structure metrics compared to previous studies.

Many alternatives for partitioning the rating data and splitting the large CF task to smaller tasks have been proposed in the literature. The proposed methods either group users together based on their social connections [47], or cluster the items together based on the ratings they share in common [48]. Finally, several methods first merge the rating information with the user social network information and then apply factorization techniques [49] or multi-way spectral clustering, using only the top few eigenvectors and eigenvalues to partition the data [50].

5.4.1.2 Parallel and Distributed Algorithms

Since the volume of data from social networks keeps growing to huge levels and many state-of-the-art collaborative filtering algorithms struggle to keep up with, it was clear to the research community that parallel and distributed system techniques could be utilized to deal with the data processing and solve scalability issues. Based on this idea, a large number of researches have been published that describe implementations of traditional collaborative filtering algorithms on parallel and distributed setups. In this context, frameworks such as OpenMP, Pthreads and Java Threads for parallel programming have been used for collaborative filtering [51]. Two of the most popular distributed frameworks, Mahout [52] and Apache Spark [46] have been widely used for high-volume data processing.

Based on the idea of distributed processing, authors in [53] proposed a Distributed Partitioned Merge (DPM) model, which is a hybrid model for large social network graphs processing. For the creation of their model they leverage the simplicity provided by Fork-Join programing and the scalability provided by Pregel framework. As reported by the evaluation findings, DPM seems to outperform both Pregel and Fork-Join in terms of recommendation time, but with a minor penalization in network usage.

Another group of research has been focused on recommender systems in decentralized, P2P environments. The peers of such networks can share profile and history information only with peers having similar interests, thus reducing the load for the recommender system and allowing a decentralized and scalable implementation [54].

The PipeCF algorithm [55, 56], has also been designed for P2P networks. The algorithm first divides the original user database into buckets which are stored in different peers and each is assigned an identifier in order to use this as a key when needed. Then PipeCF uses the information from all users in the same bucket with the active user to compute the predictions. The algorithm increases the weights for the contributions of the most similar users (unanimous amplification) and decreases the weights for users that have rated many items (significance refinement) in a process which is similar to TF/IDF weighting of terms in a text collection.

Authors in [57] created a model for predicting user to item relevance scores proposing the use of buddy-tables, which are actually tables for storing the relevance among items in ranked order (the information about the top-N relevant items). The information of these tables "follows" each item and is available among the peers of a P2P network. The buddy tables are implicitly used to create a kind of semantic layer of information in order to cluster together similar multimedia files. Every time a user starts a transaction (e.g. downloads a file), the buddy tables are updated keeping the system up-to-date. With regard to the recommendation process, given a user profile that we would like the system to create recommendation for, the buddy tables for this user have to be downloaded and the relevance scores for every item in these tables are calculated. Based on these calculations, the items that were "classified" in the top-N suggestions return to the user in the form of recommended items. Simulation experiments conducted on user logs from the Audioscrobbler community showed promising results for this P2P recommendation technique.

Another approach on dealing with the sparsity and scalability problem of the model-based Collaborative Filtering methods is the scalable clustering-based Collaborative Filtering (ACFSC), which focuses on reducing time complexity for the neighborhood creation for dealing with scalability [58]. The minimized usage of external data such as user profile and item metadata can maximize the adaptable domain. In addition, the combination of rating and external data can help on solving at some point the cold-start problem and the formation of better clusters by relocating users and items using the newly arrived ratings can increase the model coverage. Based on these policies, their architecture consists of four steps. Initially, the cluster model is created based on the user or item feature vectors. Based on this model, the top-N selected items are recommended, depending on the user's preferences. As a third step, the users' missing preferences caused by ratings matrix sparsity have to be predicted using other users and items clusters. Finally, at the learning phase, user and item feature vectors are learned to quantify the users' and items' qualitative characteristic.

There are also some clustering based methods proposed that use similarities between users and items to create clusters of users or items. Taking this further, multi-dimensional clustering can be used to cluster metadata information like user and item profile data and as a second step, the pruning of the created clusters is used to calculate the weighted average of the neighbor scores as a prediction for the user preference [59]. Another clustering based method proposed by Bellogin and Parapar [60] who used N-cut graph clustering to produce clusters with high user similarity, improving Collaborative Filtering performance but their approach lacks of coverage. The main limitation of the clustering based approaches is the higher computational cost for the cluster creation step than that for the Collaborative Filtering itself.

5.4.2 Scalable and Incremental CF Algorithms

In a common real-life scenario, a recommender is given a $i \times j$ ratings matrix where i and j is the number of Users and Items/Users/Products etc. respectively. and the system has to predict the unknown elements of the matrix. However, when the rating matrices are sparse, collaborative filtering suffers from noise in similarity calculations and results in poor recommendation quality. Latent factor analysis methods have been proposed in this case, in order to discover underlying user and item correlation and tackle the critical issue of similarity computational cost. In model-based approaches, several techniques that use the rating matrix in order to train a prediction model have been proposed, such as Principal Component Analysis (PCA), Latent Dirichlet Analysis (LDA) and Singular Value Decomposition (SVD). As stated by [4], these matrix factorization techniques are often preferred because they offer high accuracy and scalability. Factorization of the user-item ratings matrix [61] has become a quite popular solution for recommender systems after the Netflix prize competition, which indicated the ability of matrix factorization technique to achieve higher accuracy than the neighborhood-based techniques for creating item recommendations. In addition, using matrix factorization techniques allowed the use not only of explicit but implicit information as well.

The idea behind matrix factorization is the expression of both nodes of the system (i.e. users and items) into feature vectors based on patterns (latent factors) recognized in the edge list (i.e. user-to-item ratings). For the same reason, matrix factorization has been employed in many latent factor models, where recommendations are based on similarities computed over the generated item and user factors. Their ability to model different real life data and to provide good predictions made them very popular.

In order to achieve scalability in matrix factorization and keep the quality of recommendations high, the proposed methods, use overlapping partitions or decompositions of the original matrix and external knowledge (e.g. from content or from social networks) that decreases the sparsity of the original matrix. In a different direction, incremental methods use only new ratings to (re-)train an existing model.

In [62] describe how Amazon uses topic diversification algorithms to improve its recommendation. In more detail, the system that they present, uses collaborative filtering method to overcome scalability issue by generating a table of similar items offline through the use of item-to-item matrix. The system then recommends other products which are similar online according to the users' purchase history.

The idea of combining ratings and social network information has been revisited in SoRec algorithm [63] resulting in a huge sparse matrix. This time probabilistic matrix factorization has been applied in order to reduce the sparsity of the matrix. A common shared latent factor that captures both user-item rating and users' social trust led to improved predictions.

An incremental learning approach has been introduced in [64], in order to recommend high-quality videos in real-time. The factorized matrix was updated using implicit feedback from different user actions and item similarity considered additional factors, such as video type and duration. Finally, they propose the scalable implementation of their algorithm together with some optimizations to make the recommendations more efficient and accurate, including the demographic filtering and demographic training.

The use of incremental algorithms, can also improve the scalability of model-based recommender systems, since only the newer information is used to train the model. Luo et al. [65] used Regularized Matrix Factorization to create an incremental Recommender System. The Recommender System that they proposed creates an initial model with the given parameters and allows their model to be incrementally trained, meaning that their model is incrementally updated as new ratings arrive. These kind of approaches provide real improvement of the quality of recommendations solving partially the problem of sparsity of the rating matrices but actually deteriorate the scalability issue.

In the compact latent factor model proposed in [66], the item-scoring function was trained periodically, to reduce the training overhead. Authors introduced a buffer mechanism to retrieve the data incrementally and compared to traditional learning, achieved better scalability, since their model updated only when the number of data instances in the current buffer is sufficient rather than each time a rating was added.

Finally, in [67] authors exploit the parallel computing platform and application programming interface (API), known as CUDA, which improved the performance, exploiting the graphics processing unit (GPU) capabilities in high processing and propose a CUDA based matrix factorization library called CuMF that uses the method of alternate least squares (ALS) to implement matrix factorization in large-scale datasets. This proposed method aims on increasing performance in both single and multiple GPUs. Some key features are the leverage of the GPU memory structure and hierarchy in order to provide easy access to sparse data and the minimization of the communication costs by using data and model parallelism.

5.4.3 Scalability of Deep Learning Solutions in Recommender Systems

Another way to confront, sparsity, cold-start, scalability and other issues is the use of hybrid filtering. This approach involves the combination of different techniques for creating recommendations, trying to improve the accuracy of the predictions [68, 69], while leveling out individual method weaknesses [70]. They can be classified based on their operations into weighted hybrid, mixed hybrid, switching hybrid, feature-combination hybrid, cascade hybrid, feature-augmented hybrid and meta-level hybrid [71].

The rise of deep learning techniques also affected the recommender systems. Towards this direction, Peska and Trojanova [72] used the visual descriptors of the deep neural networks for creating item-based recommendations as an evaluation scenario for creating a photo lineup assembling task. This is a great example of how broad is the utility of deep learning and recommendation processes in wide areas of research.

Many current researches on deep learning for recommendation systems address a variety of potentials open for discussion. In this context, Smirnova and Vasile [73], address the limitation of not using implicit information from the user profile, such as the timestamp of a user-item transaction or the user's inactivity time interval, by the Recurrent Neural Networks. Based on this, authors propose a class of RNN, called Contextual Recurrent Neural Networks (CRNNs), that uses the contextual information of the network both in the input and output of the CRNN and rectifies the behavior of the neural network in order to produce item predictions. Based on the experimental results, using YooChoose dataset and a proprietary dataset consisted by user browsing and purchasing activity on various e-commerces, this approach can achieve noteworthy results against sequential and nonsequential state of the art models.

Tackling the problem of boosting a recommender system's performance, Chatzis et al. [74] proposed the creation of a machine learning model that can derive hidden or implicit information from the sparse user session data. This idea is typically based on traditional RNN approaches which define the state-of-the-art in this domain. Towards the same direction, Zanotti et al. [75] collect and combine data from multiple sources such as user-item ratings, user-item reviews and item descriptive data in the effort of forming rich distributed representations and enhancing this information in the classic RNN systems. The further goal of this method is to try to boost the prediction of the user to item actual score.

Based on the experimental findings of the approaches mentioned above, all of these enhancements can help recommender systems achieve better results with the use of deep learning methods such as neural networks. Based on the survey of research on deep learning and recommender systems, the pinpoint that need attention is that even though there is plenty of research in deep learning for these kind of purposes and many approaches seem to outperform traditional techniques for providing higher results of recommendations in many cases, all these approaches mainly target on improving the actual recommendation algorithm scores rather than solving the scalability issues. In that sense, we consider that there is plenty of research interest towards this area and the need for that seems necessary.

5.5 Conclusions

Dealing with the problem of creating real-life recommendations in large-scale and sparse networks can be a challenging task both regarding scalability, data sparsity and recommendation quality of course.

The major challenges in RS are the scalability, diversity and the long tail, sparsity and the cold-start problem. In addition, content-based filtering faces the problem of content analysis in the cases where the sets of items lack of a respectively sufficient set of features. Finally another challenge is the combination of collaborative filtering and content-based filtering approaches in hybrid filtering cases. A solution about the scalability and data management problem in CF-recommenders, can be the idea of splitting the (social) graph into sub-graphs as long as the use of parallel and distributed algorithms. Matrix factorization is another solution for scalable and incremental CF-algorithms. Finally a way to confront, sparsity, cold-start, scalability and other related problems is the use of hybrid filtering techniques especially based on deep learning.

Since many state-of-the-art collaborative filtering algorithms struggle to keep up with the three main characteristics of big data, that could describe the current state in many large scale social networks, Volume, Velocity, Variety -the also known as three Vs- it was clear that parallel and distributed technologies could and would offer potential solutions to some of these issues. In this context, many research works have proposed the implementation of existing recommendations algorithms on parallel and distributed systems. Many modern recommender systems are based in partitioning or clustering methods trying to deal with the scalability, but a trend in using parallel and distributed systems is observed. Most of these approaches try to distribute the data over a cluster or parallelize the classic collaborative filtering algorithms used in traditional techniques to deal with the scalability problem.

The last few years the field of deep learning seems to gain a lot of research focusing in the use of techniques like Recurrent Neural Networks for processing data and creating very accurate recommendations. However, the main point of concern is that although these methods outperform traditional approaches in many cases, still they try to encounter the issues of sparsity and cold-start recommendations and not the problem of scalability. Based on this, there seems to be plenty of open space for research towards this direction.

References

1. Symeonidis, P., Ntempos, D., & Manolopoulos, Y. (2014). *Recommender systems for location-based social networks*. Springer Science & Business Media.
2. Nepali, R., & Wang, Y. (2014). *SocBridge: Bridging the gap between online social networks*.
3. Chakradhar, S., & Raghunathan, A. (2010). IEEE: Best-effort computing: Re-thinking parallel software and hardware. In *2010 47th ACM/IEEE on Design Automation Conference (DAC)* (pp. 865–870).
4. Ricci, F., Rokach, L., & Shapira, B. (2011). *Introduction to recommender systems handbook* (pp. 1–35). Springer.
5. Chen, H.-H., Ororbia, I., Alexander, G., & Giles, C. (2015). ExpertSeer: A Keyphrase Based Expert Recommender for Digital Libraries. arXiv:1511.02058.
6. Chen, H.-H., Gou, L., Zhang, X., & Giles, C. (2011). ACM: Collabseer: a search engine for collaboration discovery. In *Conference Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries* (pp. 231–240).
7. Felfernig, A., Isak, K., Szabo, K., & Zachar, P. (1999). The VITA financial services sales support environment. In *Conference Proceedings of the National Conference on Artificial Intelligence* (Vol. 22, p. 1692). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press.
8. Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., & Zadeh, R. (2013). ACM: Wtf: The who to follow service at twitter. In *Conference Proceedings of the 22nd International Conference on World Wide Web* (pp. 505–514).

9. Montaner, M., López, B., & De La Rosa, J. (2003). A taxonomy of recommender agents on the internet. *Artificial Intelligence Review, 19*(4), 285–330.
10. Lops, P., De Gemmis, M., & Semeraro, G. (2011). *Content-based recommender systems: State of the art and trends* (pp. 73–105). Springer.
11. Su, X., & Khoshgoftaar, T. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence* (Vol. 4).
12. Burke, R.: Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370.
13. Sardianos, C., Varlamis, I., & Eirinaki, M. (2017). Scaling collaborative filtering to large-scale bipartite rating graphs using lenskit and spark. In *BigDataService* (pp. 70–79).
14. Park, D., Kim, H., Choi, I., & Kim, J. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications, 39*(11), 10059–10072.
15. Oard, D., Kim, J., others, Menlo Park, CA: AAAI Press: Implicit feedback for recommender systems. In *Conference Proceedings of the AAAI workshop on recommender systems* (pp. 81–83).
16. Carrer-Neto, W., Hernández-Alcaraz, M., Valencia-García, R., & García-Sánchez, F. (2012). Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications, 39*(12), 10990–11000.
17. Arazy, O., Kumar, N., & Shapira, B. (2009). Improving social recommender systems. *IT professional, 11*(4).
18. Li, Y.-M., Liao, T.-F., & Lai, C.-Y. (2012). A social recommender mechanism for improving knowledge sharing in online forums. *Information Processing & Management, 48*(5), 978–994.
19. Siersdorfer, S., & Sizov, S. (2009). ACM: Social recommender systems for web 2.0 folksonomies. In *Conference Proceedings of the 20th ACM Conference on Hypertext and hypermedia* (pp. 261–270).
20. Hossain, L., & Fazio, D. (2009). The social networks of collaborative process. *The Journal of High Technology Management Research, 20*(2), 119–130.
21. Perugini, S., Gonçalves, M., & Fox, E. (2004). Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems, 23*(2), 107–143.
22. Melville, P., & Sindhwani, V. (2011). *Recommender systems* (pp. 829–838). Springer.
23. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). ACM: Item-based collaborative filtering recommendation algorithms. In *Conference Proceedings of the 10th International Conference on World Wide Web*, pp. 285–295 (2001)
24. Cotter, P., Smyth, B.: Ptv: Intelligent personalised tv guides. In *AAAI/IAAI* (pp. 957–964).
25. Strub, F., Gaudel, R., & Mary, J. (2016). ACM: Hybrid recommender system based on autoencoders. In *Conference Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (pp. 11–16).
26. Zhao, W., Li, S., He, Y., Chang, E., Wen, J.-R., & Li, X. (2016). Connecting social media to e-commerce: Cold-start product recommendation using microblogging information. *IEEE Transactions on Knowledge and Data Engineering, 28*(5), 1147–1159.
27. Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications, 69*, 29–39.
28. Wang, H., Wang, N., & Yeung, D.-Y. (2015). ACM: Collaborative deep learning for recommender systems. In *Conference Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1235–1244).
29. Liu, J., Wu, C., Springer: Deep Learning Based Recommendation: A Survey. In *International Conference on Information Science and Applications* (pp. 451–458).
30. Schein, A., Popescul, A., Ungar, L., & Pennock, D. (2002). ACM: Methods and metrics for cold-start recommendations. In *Conference Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 253–260).
31. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28*(1), 11–21.

32. Patel, K., Thakkar, A., Shah, C., & Makvana, K. (2016). Springer: A state of art survey on shilling attack in collaborative filtering based recommendation system. In *Conference Proceedings of First International Conference on Information and Communication Technology for Intelligent Systems* (Vol. 1, pp. 377–385).
33. Hurley, N., O'Mahony, M., & Silvestre, G. (2007). Attacking recommender systems: A cost-benefit analysis. *IEEE Intelligent Systems, 22*(3).
34. Shyong, K., Frankowski, D., & Riedl, J. (2006). others: Do you trust your recommendations? *An exploration of security and privacy issues in recommender systems* (pp. 14–29). Springer.
35. McSherry, F., & Mironov, I. (2009). ACM: Differentially private recommender systems: building privacy into the net. In *Conference Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 627–636).
36. Koren, Y. (2008). ACM: Tutorial on recent progress in collaborative filtering. In *Conference Proceedings of the 2008 ACM Conference on Recommender Systems* (pp. 333–334).
37. Felfernig, A., & Gula, B. (2006). IEEE: An empirical study on consumer behavior in the interaction with knowledge-based recommender applications. In *The 3rd IEEE International Conference on E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services* (pp. 37–37).
38. Agrawal, D., & Aggarwal, C. (2001). ACM: On the design and quantification of privacy preserving data mining algorithms. In *Conference Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (pp. 247–255).
39. Nayebzadeh, M., Moazzam, A., Saba, A., Abdolrahimpour, H., & Shahab, E. (2017). *An investigation on social network recommender systems and collaborative filtering techniques*. arXiv: 1708.00417.
40. Agrawal, D., El Abbadi, A., Das, S., & Elmore, A. (2011). Springer: Database scalability, elasticity, and autonomy in the cloud. In *International Conference on Database Systems for Advanced Applications* (pp. 2–15).
41. Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review, 44*(2), 35–40.
42. Maqsood, T., Khalid, O., Irfan, R., Madani, S., & Khan, S. (2016). Scalability Issues in Online Social Networks. *ACM Computing Surveys (CSUR), 49*(2), 40.
43. Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132.
44. Lloyd, W., Freedman, M., Kaminsky, M., & Andersen, D. (2014). Don't settle for eventual consistency. *Communications of the ACM, 57*(5), 61–68.
45. Pujol, J., Erramilli, V., Siganos, G., Yang, X., Laoutaris, N., Chhabra, P., et al. (2010). The little engine (s) that could: scaling online social networks. *ACM SIGCOMM Computer Communication Review, 40*(4), 375–386.
46. Shanahan, J., & Dai, L. (2015). ACM: Large scale distributed data science using apache spark. In *Conference Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2323–2324).
47. Pham, M., Cao, Y., Klamma, R., & Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of UCS, 17*(4), 583–604.
48. O'Connor, M., Herlocker, J., & Berkeley, U. (1999). Clustering items for collaborative filtering. In *Conference Proceedings of the ACM SIGIR Workshop on Recommender Systems* (Vol. 128).
49. De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2011). IEEE: Improving recommendation quality by merging collaborative filtering and social relationships. In *2011 11th International Conference on Intelligent Systems Design and Applications (ISDA)* (pp. 587–592).
50. Symeonidis, P., Iakovidou, N., Mantas, N., Manolopoulos, Y. (2013). From biological to social networks: Link prediction based on multi-way spectral clustering. *Data & Knowledge Engineering, 87*, 226–242.
51. Eirinaki, M., Gao, J., Varlamis, I., Tserpes, K. (2017). Recommender systems for large-scale social networks: A review of challenges and solutions. *Future Generation Computer Systems, 78*, 412–417.

52. Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in action: Manning Shelter Island*.
53. Corbellini, A., Godoy, D., Mateos, C., Schiaffino, S., & Zunino, A. (2017). DPM: A novel distributed large-scale social graph processing framework for link prediction algorithms. *Future Generation Computer Systems.*
54. Shavitt, Y., Weinsberg, E., & Weinsberg, U. (2010). ACM: Building recommendation systems using peer-to-peer shared content. In *Conference Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 1457–1460).
55. Han, P., Xie, B., Yang, F., & Shen, R. (2004). A scalable P2P recommender system based on distributed collaborative filtering. *Expert Systems with Applications, 27*(2), 203–210.
56. Han, P., Xie, B., Yang, F., Wang, J., & Shen, R. (2004). Springer: A novel distributed collaborative filtering algorithm and its implementation on p 2p overlay network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 106–115).
57. Wang, J., Pouwelse, J., Lagendijk, R., & Reinders, M. (2006). ACM: Distributed collaborative filtering for peer-to-peer file sharing systems. In *Conference Proceedings of the 2006 ACM Symposium on Applied Computing* (pp. 1026–1030).
58. Lee, O.-J., Hong, M.-S., Jung, J., Shin, J., & Kim, P. (2016). Adaptive collaborative filtering based on scalable clustering for big recommender systems. *Acta Polytechnica Hungarica, 13*(2), 179–194.
59. Li, X., & Murata, T. (2012). IEEE Computer Society: Using multidimensional clustering based collaborative filtering approach improving recommendation diversity. In *Conference Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03* (pp. 169–174).
60. Bellogin, A., & Parapar, J. (2012). ACM: Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Conference Proceedings of the Sixth ACM Conference on Recommender Systems* (pp. 213–216).
61. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8).
62. Ziegler, C.-N., McNee, S., Konstan, J., & Lausen, G. (2005). ACM: Improving recommendation lists through topic diversification. In *Conference Proceedings of the 14th International Conference on World Wide Web* (pp. 22–32).
63. Ma, H., Yang, H., Lyu, M., & King, I. (2008). ACM: Sorec: social recommendation using probabilistic matrix factorization. In *Conference Proceedings of the 17th ACM Conference on Information and Knowledge Management* (pp. 931–940).
64. Huang, Y., Cui, B., Jiang, J., Hong, K., Zhang, W., & Xie, Y. (2016). ACM: Real-time video recommendation exploration. In *Conference Proceedings of the 2016 International Conference on Management of Data* (pp. 35–46).
65. Luo, X., Xia, Y., & Zhu, Q. (2012). Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems, 27*, 271–280.
66. Liu, C.-L., & Wu, X.-W. (2016). Large-scale recommender system with compact latent factor model. *Expert Systems with Applications, 64*, 467–475.
67. Tan, W., Cao, L., & Fong, L. (2016). ACM: Faster and cheaper: Parallelizing large-scale matrix factorization on gpus. In *Conference Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing* (pp. 219–230).
68. Göksedef, M., & Gündüz-Ö\ugüdücü, S. (2010). Combination of Web page recommender systems. *Expert Systems with Applications, 37*(4), 2911–2922.
69. Mobasher, B. (2007). Recommender Systems. *KI, 21*(3), 41–43.
70. Al-Shamri, M., & Bharadwaj, K. (2008). Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. *Expert Systems with Applications, 35*(3), 1386–1399.
71. Mican, D., & Tomai, N. (2010). Springer: Association-rules-based recommender system for personalization in adaptive web-based applications. In *International Conference on Web Engineering* (pp. 85–90).
72. Peska, L., & Trojanova, H. (2017). *Towards recommender systems for Police Photo Lineup*. arXiv:1707.01389.

73. Smirnova, E., & Vasile, F. (2017). *Contextual sequence modeling for recommendation with recurrent neural networks*. arXiv:1706.07684.
74. Chatzis, S., Christodoulou, P., & Andreou, A. (2017). *Recurrent latent variable networks for session-based recommendation*. arXiv:1706.04026.
75. Zanotti, G., Horvath, M., Barbosa, L., Immedisetty, V., & Gemmell, J. (2016). ACM: Infusing collaborative recommenders with distributed representations. In *Conference Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (pp. 35–42).