New Insights Towards Developing Recommender Systems

Abstract

Promoting recommender systems in real-world applications requires deep investigations with emphasis on their next generation. This survey offers a comprehensive and systema-tic review on recommender system development life cycles to enlighten researchers and practitioners. The paper conducts statistical research on published recommender systems indexed by *Web of Science* to get an overview of the state of the art. Based on the review ed findings, we introduce taxonomies driven by the following five phases: initiation (arch-itecture and data acquisition techniques), design (design types and techniques), develop-ment (implementation methods and algorithms), evaluation (metrics and measurement techniques) and application (domains of applicatio ns). A layered framework of recomm-ender systems containing market strategy, data, recommender core, interaction, security and evaluation is proposed. Based on the framework, the existing advanced humanized techniques emerged from computational intelligence and some inspiring insights from computational economics and machine learning are provided for researchers to expand the novel aspects of recommender systems.

Keywords: Taxonomy, systematic review, computational intelligence, recommendation techniques, similarity computation algorithms, evaluation.

1. Introduction

Our global information society is increasingly producing large amounts of data which makes finding useful and relevant information at the right moment difficult. We face, daily, various options about services and products that need to be filtered

based on our preferences. Recommender systems have emerged to provide a means for handling vast amount of data on the web by retrieving the most relevant information in a customized manner. These systems aim to provide personalized models by gathering user activities and/or item data to assist users per their preferences expressed either explicitly or implicitly. Recommender systems generate a list of items (or people) to be recommended to the users.

Historically, the area of recommender systems was part of data mining and information filtering. Later, in the 90s, it has been recognized as a full-fledged research area. Currently, major companies such as Amazon, Netflix, Launch, Google, YouTube and Facebook are heavily using and relying on recommender systems to sell their products and services by recommending the most relevant items to the users, leading to a significant increase of revenue. Due to its economical role, Netflix announced the Netflix Grand Prize (1 Million US Dollars), an open competition on the best collaborative recommender system to predict the ratings of the films based on the users' preferences. BellKor' Pragmatic Chaos won the Netflix 2009 Grand Prize by providing prediction at only 10.06% of accuracy [1]. Another successful example is the Chris Anderson's book, "Touching the Void", that became the bestselling book on Amazon a few years after its publication thanks to the collaborative recommender system employed on Amazon website [1].

Among the earliest published surveys of recommender systems, a classification of recommender system techniques and an architecture for hybrid recommender systems have been provided by Burke [2, 3] in 2002 and 2007. In 2005, Adomavicius et al. [4] conducted another survey on limited techniques including content-based, collaborative and hybrid filtering. Following the discussion of few related challenges, the authors suggested the incorporation of the contextual

information and support for multi-criteria ratings. In 2011, Shani et al. [5] studied the efficiency and effectiveness of recommender systems and proposed a complete list of measures for evaluating their performance in theory and application. Though, the use of these measures by researchers and practitioners has not been explored yet. In an attempt to study the algorithmic aspect of recommender systems, a survey on recommendation algorithms and user satisfaction has been conducted by Konstan et al. [6] in 2012. However, the survey does not cover a variety of algorithms and lacks a comprehensive classification of the covered ones. Bobadila et al. (2013) [7] provided a through survey on recommender system's theory with specific attention to collaborative filtering techniques and algorithms. The authors summarized the evolution of recommender systems in the context of the recent web advancements (Web 2.0 and 3.0). In a recent study by Lu et al. [8] in 2015, eight different applications of recommender systems were identified, namely e-government, e-business, e-commerce/e-shopping, e-library, e-learning, e-tourism, e-resource services and e-group activities. Although recommender systems are receiving great interest in business and real life applications, further research and development are still needed for these systems to be efficiently applied in complex settings and many challenging issues are yet to be addressed [9].

While the existing reviews and surveys on recommender systems focus mainly on relevant techniques and algorithms, our paper tries to draw a broader picture and aims to illustrate the evolvement of traditional techniques to advanced and intelligent methods to guide researchers and practitioners with an insight vision towards the next generation of such systems. To avoid loss of generality and provide new advancements concealed from past surveys, the paper considers

various ways of recommender system develop-ents and analyzes their different advantages and disadvantages to deliver quality and effective systems. In fact, the paper provides a comprehensive review on how to engineer recommender systems from the system development life cycle viewpoint. It also presents taxonomies of recommender systems during the following five phases: initiation, design, development, evaluation and application. Furthermore, the paper discusses initiatives and primary approaches and introduces future research opportunities from different perspectives ranging from engineering to computer science, mathematics, business and computational economics.

Another distinct feature of this paper is its investigation of existing research papers guided by the aim to extract some relevant and informative statistical data about the venues of published work and active groups in this domain to get an overview of the state of the art and motivate the significance of this area of research. This research contributes as follows:

1. Introducing comprehensive taxonomies driven by different ways of developing recom-mender systems.
2. Discussing relevant and important techniques and algorithms used to develop these systems while highlighting the related challenges.
3. Exploring evaluation techniques and potential metrics.
4. Comparing the alternative approaches and techniques to expose the strengths and weaknesses.
5. Investigating how computational intelligence contributed to improve recommendation-s.
6. Proposing new insights into this field.

The rest of the paper is organized as follows. The methodology and process of developing taxonomies are discussed in Section 2. Section 3 presents statistics of research studies conducted in the domain of recommender systems. Different taxonomies of the recommender system's life cycle are provided in Section 4. New insights and future research opportunities to develop the next generation of recommender systems are identified and discussed within a proposed layered framework in Section 5. Finally, Section 6 concludes the paper.

2. Review Methodology and Taxonomy Development Process

The paper is organized in two parts. First, the status of research publications in recommender systems is presented. The *Web of Science* database by Thomson Reuters has been selected for this statistical investigation. Several keywords including, but not limited to "recommender system", "recommender technique", "recommendation agent", "recommendation system", "recommendation method", "collaborative filtering", "content-based", "hybrid recommender", "knowledge-based recommender", "user modeling" and "user profiling" have been explored in the date range between 2005 and 2015. The retrieved papers are analyzed in Section 3. Second, a systematic survey was performed on recommender systems. Unlike the first part, for the survey part of the paper, we did not limit our study to any time to ensure its comprehensiveness. The following steps have been conducted in our systematic review:

Search: a preliminary investigation was performed on "*Scopus*" and "*Web of Science*" databases with the same keywords mentioned above. Specific attention was given to books and survey papers.

Paper selection: when selecting a paper among the retrieved papers, we considered inclusion and exclusion criteria. All the retrieved survey papers and books were

considered for the review. For a research paper to be included in our survey, it must introduce a new approach, technique, algorithm, evaluation measure or application. Otherwise, the paper was opted out. Among the selected papers, those using methods having very low popularity were eliminated.

Paper review: Taxonomy development is not an easy task. In this paper, we follow the taxonomy development steps in information systems provided by Nickerson et al. as depicted in Figure 1 [10]. As argued by the authors, being concise, sufficiently inclusive, comprehensive and extensible are the four quality characteristics for a desired taxonomy. The taxonomy development task is broken into the following steps:

1- Investigate a subset of objects which should be classified; they are probably known by the developer or easily accessible through a literature review.

2- Identify general characteristics of the collected objects and investigate in which ways they are similar or what distinguishes them from each other.

3- Assemble the characteristics into dimensions to form an initial taxonomy; each dimension consists of a set of characteristics that are mutually exclusive and collectively extensive.

4- Conceptualize new characteristics and dimensions that might not have been identified or present previously. In this process, some dimensions or characteristics might be deduced or even combined.

5- Examine the objects with the new perspectives of characteristics and dimensions, then revise the taxonomy and create a new version.

6- Discover missing objects in the taxonomy after it is completed and used.

7- Design new objects with the missing characteristics.

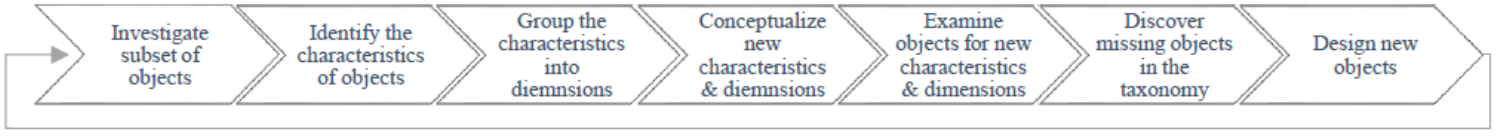


Figure 1. Steps of taxonomy development

3. Status of Recommender Systems Research

Overall, 5333 publications were retrieved from the *Web of Science* database which were related to recommender systems. Figure 2 presents the number of publications and their citations in each year since 2005 until May, 2015. The trend of the number of publications is showing a slight increase since 2013 despite two periods of increase. Meanwhile, citations are expected to rise as Figure 2 is illustrating.

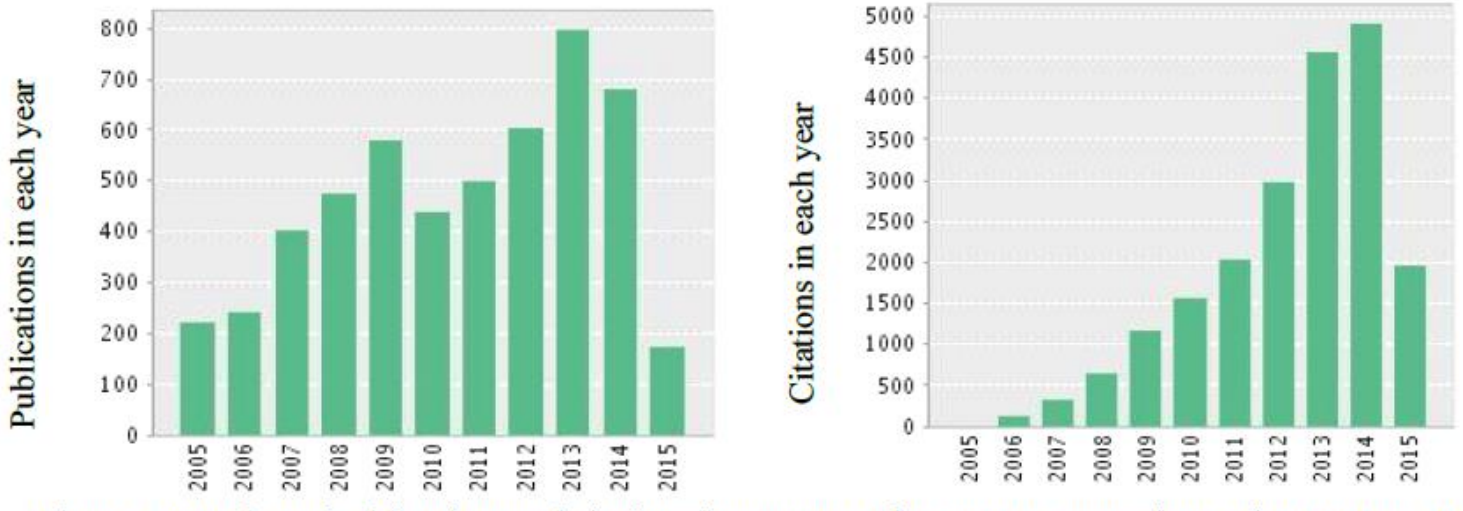


Figure 2. Number of publications and citations in recommender systems research area (up to May, 2015)

From the retrieved publications, 58% are Conference papers and about 41% are published in academic journals. Very few publications are review studies or books as shown in Table 1. Even though the dominant language is English, a considerable number of journal papers are published in other languages.

Document type	Number of publications	English language	Non-English Language
Conference	3093	3079	14
Journal paper	2177	1865	312
Review	38	38	0
Book	5	5	0
Patent/other	20	14	6

Table 1. Types of retrieved research documents in recommender systems

As shown in Figure 3, the country that contributed the most in this field is China. China has published 1026 research articles followed by the United States that contributed with 592 publications. This is not surprising since according to a report of science watch by Thomson Reuters [11], the USA and China, in this order, have the highest amount of papers and citations in all research fields. Then Spain, South Korea, Taiwan, Germany, Italy, Japan, India and Canada are the major contributors to this research field. Spain, Germany, Italy and Japan are among the top 10 places in the overall number of publications and citations subject wide as well. South Korea, India and Taiwan are listed among the top 20 in publications and citations. However, to have a fair judgment of countries' publications, we need to consider multiple criteria such as science funding, population size, GDP, education levels, facilities and language advantages which are beyond the scope of this study.

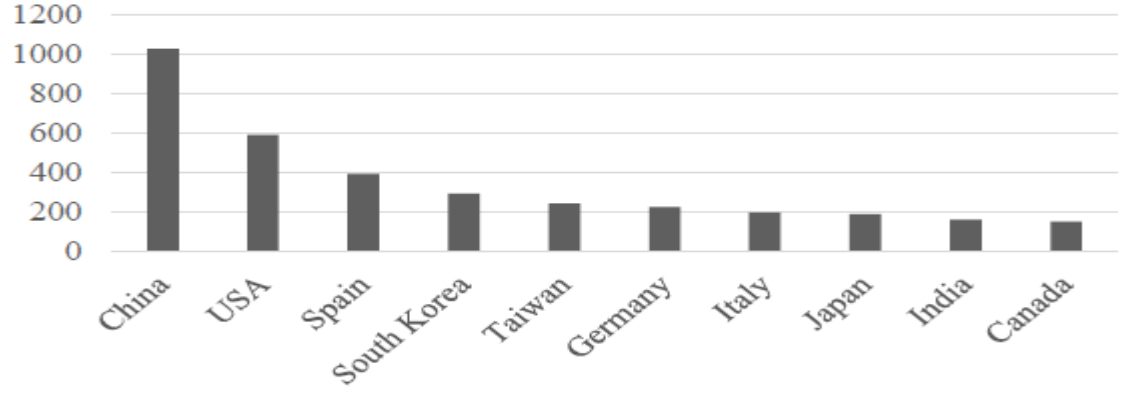


Figure 3. Top countries in recommender systems publications

As expected, the National Natural Science Foundation of China had the largest investment in the research conducted on recommender systems. The European Union, the National Science Foundation of USA, and the Swiss National Science Foundation are the major funding agencies which contribute to recommender systems advances. The fifth position is held by another Chinese agency. The exact numbers of research papers funded by these agencies are reported in Figure 4. Since recommender systems research is interdisciplinary, we investigated the research fields of the retrieved publications to understand their content relevancy. As expected, computer science and computer engineering are the main research fields of the considered publications. Social media is receiving the highest attention from researchers, with 371 publications. Particularly, many researchers have used social network-based techniques [12]. Moreover, several research proposals have been produced on recommendation algorithms using mathematical models; making mathematics the second most used field (218 of total publications). Recommender systems have strongly contributed in the business evolvement of multiple companies in the way of marketing or even providing public services [6].

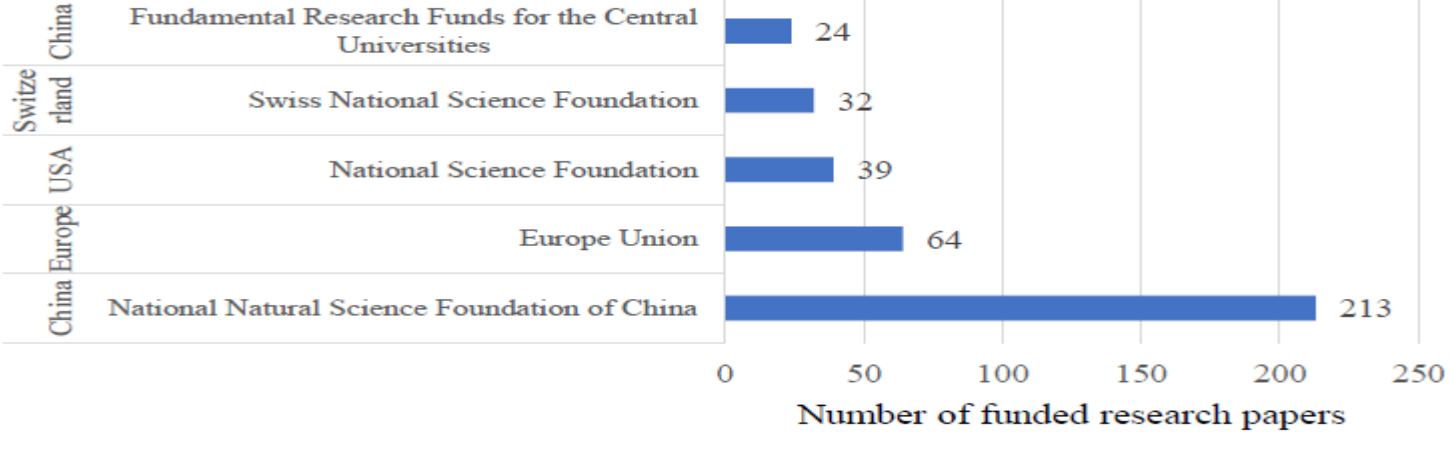


Figure 4. Top-five funding agencies for recommender systems research

Therefore, business and economic considerations occupy the third place of the most searched fields with 198 publications. Education and health have also tried to get advantages of recommender systems in providing their 5 services which were cited in 139 and 62 publications respectively. Later, we will introduce the other applications of recommender systems which have potential and have not been well utilized yet.

4. Recommender System Taxonomy

4.1) Initiation

The first step to develop a recommender system is to think of the right approaches about the system and available data. Figure 5 presents four different approaches which might be considered as the initial task for the development of a recommender system. They include what kind of data we have or we need, how to obtain the data and what are the system structure requirements.

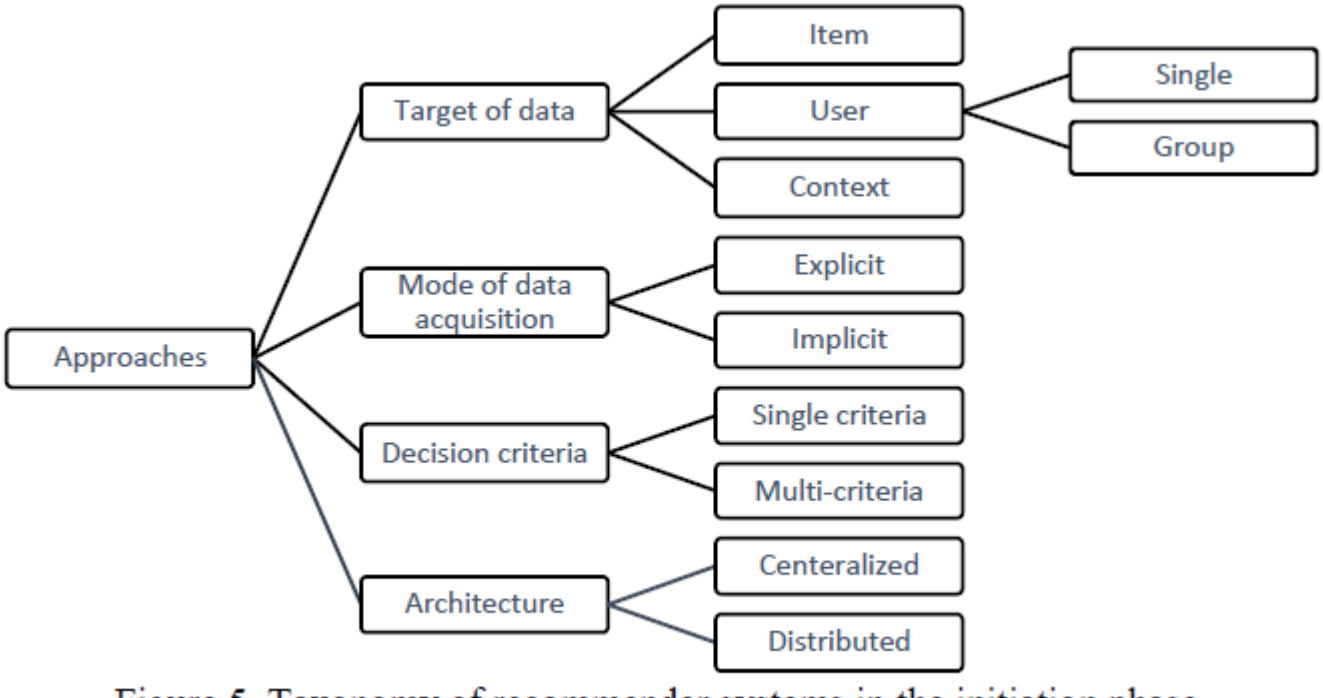


Figure 5. Taxonomy of recommender systems in the initiation phase

Target of data

The data target user, item or context. Most of the recommender systems applied in e-commerce follows the user-centered approach [13]. This approach determines similarities between different users and stores user-to-user relations. From the sales point of view, this approach is advantageous since it collects valuable user information which can be used in customer relationship management. On the other hand, it is not necessarily preferred on the user side due to privacy concerns, which may impede user participation. When it comes to applications with strong privacy affairs, item-centered recommender systems are a better choice [14]. These systems observe users' usage of items anonymously and keep the data as item-to-item relations. There is another complementary approach which collects contextual data along with the above-mentioned approaches. In recommender systems, contextual data is considered 6 as any additional data which has a direct impact on the relevance of recommendations such as time, location, nearby people or objects.

Recommendations may be provided for a single user or may target a group of users, such as recommending a travel package for a family. Most of recommender systems consider an individual user and deal with the optimal options for a single user. However, recommending to a group of users would be more complicated since it should combine users' models and be optimal for all users [15]. Providing recommendations to users requires certain type of identification from those users. User observation or user explicit input is a part of this identification. As much as the system acquires information regarding the user, the recommendations would be more personalized. It is worth mentioning that there can be a case where the system does not have any information regarding a specific

user. In this case, the system would generate non-personalized recommendations for the unknown user [16].

Mode of acquisition

The type of the data stored in a database may vary in different aspects such as ratings, items features and content, registration information of users, social relationships and so on. If the required data is not currently available in the database, then it should be collected in two different ways [17]:

 Explicit: the input data comes from factual data about items or users (e.g., item features, user demographic data and time) or a direct user feedback, such as ratings to items made by users.

 Implicit: the input data is based on the behavioral usage such as a user's purchase behavior, browser session location, number of times a user has heard a song or detection of user's feeling about the song [18].

The explicit method has the advantage of simplicity, but may fail in cognitive measures to catch the users' feelings about items. The implicit method does not require a direct user involvement, although bias is likely to happen such as phone call interruption while reading.

Decision criteria

In a recommender system, the items of interest and the user preferences are represented in the forms of singular or multiple attributes. Particularly in the systems where recommendations are based on the opinions of others, it is crucial to take into consideration the multiple criteria that affect the users' opinions to make effective recommendations. Researchers have used Multi Criteria Decision Making (MCDM) methods to facilitate the process of recommendation creation [19]. This is specifically

practical when there is a group of users. The commonly utilized techniques for MCDM are 1) finding Pareto optimal or multiple criteria linear combinations; and 2) reducing the criteria to a single-criterion solution by utilizing the most important criterion or using one criterion at each time [20].

Architecture

Two types of recommender system architectures are distinguished: centralized and distributed. Centralized means that the recommender system is located in one particular place. In a distributed architecture, the system components are distributed over several locations. Peer-to-peer architecture is an example of distributed architecture. Distributed algorithms are not as accurate as their centralized versions that have complete user profile information [21]. However, disaggregation of user information in distributed settings can solve the privacy issues to a large extent. Distributed recommender systems are not experienced with yet in real world applications compared to the centralized ones [22]. The cooperation of users is very important and necessary to make the system run properly.

4.2) Design

The next step is to specify a technique depending on the application. Different techniques are shown in Figure 6. There are various classifications of recommender system techniques, and the most recognized one is defined by Burke as followed [3]:

Content-based: in this technique, the recommender system analyzes a set of descriptions of items that a user has rated in the past. Then, it builds a user model or profile of the users' interests according to the features of the rated objects and matches the attributes of the profile against the attributes of a content of an object. Thus, the system learns to recommend items similar to the ones liked in the past. For example, if a comedy movie is

given a positive rating by a user, then the system learns to recommend other movies belonging to the same category.

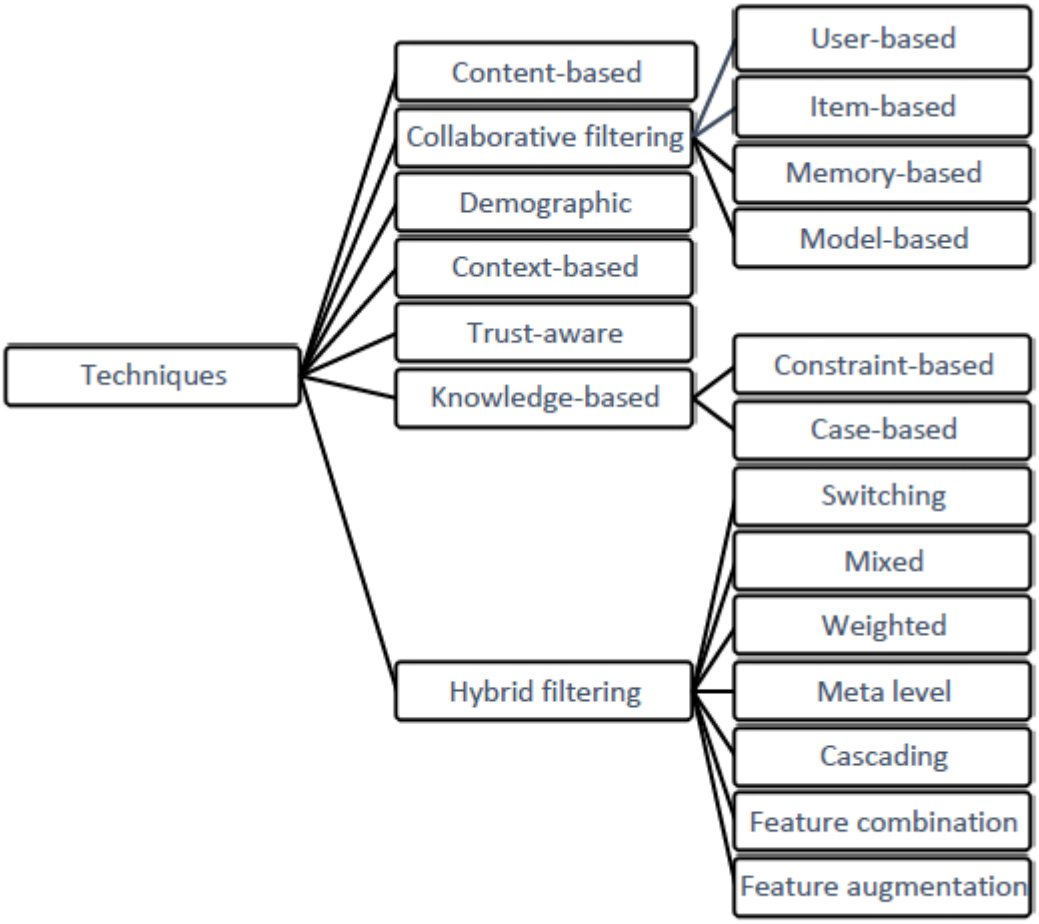


Figure 6. Taxonomy of recommender systems in the design and development phases

Collaborative filtering: collaborative techniques are the most widely known and used in recommender systems. This approach recommends items to the user that were liked in the past by other users with similar tastes. The recommendations are produced based on ratings or usage patterns without any need of vital information about users or items. The following four filtering approaches are discussed in collaborative filtering:

User-based collaborative filtering measures the correlation between pairs of users. This method has been adopted in making high quality predictions and recommendations, but it failed in practice for online applications. It performed too slowly to process hundreds of thousands or millions of users [13].

 Item-based collaborative filtering is an alternative method, assigning correlations between items and recommending items with high similarity to the set of items already rated by the user. The item-based algorithms tend to be faster in terms of online response time than the user-based algorithms, specifically, if the item relationships are pre-computed. The item-based algorithm, which also fits nicely into unary rating sets, quickly became popular in commercial applications [13]. Unary datasets are those with positive or no information at all, for instance sales data.

 Memory-based systems work only with the matrix of user-item ratings and use any rating generated before the referral process. They mostly use similarity metrics to measure the distance between two users or two items, based on each of their ratios. Memory-based methods suffer from scalability problems, since they need to process the whole data to make a single prediction, which requires high computing resources and makes the process time consuming [23].

 Model-based systems create a model using the obtained information to generate the recommendations. Model-based systems are time consuming in the preprocessing step, but once the model is generated, the recommendations can be generated instantly. Model-based systems have disadvantages as well. Some of the models are very complex as they should estimate multi-dimensional parameters. This leads to high sensitivity in terms of data changes. There is a risk of mismatching the model with the data, which causes irrelevant recommendations, so not any theoretical model has the potential to be applied in real world applications [24].

Demographic: demographic recommender systems classify users based on their personal attributes to provide recommendations per demographic profiles. The underlying theory is that different demographic niches should receive their own specific offers. This type of

recommender systems acts somehow similar to content-based, but the benefit over this approach is that unlike collaborative and content-based techniques, it does not necessarily need user ratings history [25].

Knowledge-based: this approach focuses on knowledge sources that are not revealed by content-based and collaborative filtering techniques. It generates recommendations based on specific domain knowledge of the users' requirements, the items' features and how these features can meet users' needs and preferences. This technique tends to act better than the other techniques in the beginning of their employment as they do not use any ratings. However, in order to keep this superiority, they should be equipped with learning components to exploit the human/computer interaction logs [2]. There are two schemes for knowledge-based recommendations:

- Case-based: it uses similarity metrics to retrieve items similar to users' needs.
- Constraint-based: it exploits a set of recommendation rules to find out the items which satisfy the users' requirements.

Both approaches are similar in terms of the recommendation process: the user must specify the requirements, and the system tries to identify a solution. Some studies investigated *utility-based* recommender systems which can be inferred as a specific type of knowledge-based recommender system [26]. Generally, utility-based recommendation uses a constraint-based approach and sometimes it combines it with case-based methods .Utility-based recommendations compute the utility of each item for a specific user. It uses features of items as background data, produces utility functions of items from users to explain user preferences, and utilizes the function to rank items for a user.

Context-based: it focuses on additional contextual information including time, location, wireless sensors and so on. The contextual data may be collected using explicit and

implicit feedback or data mining techniques [17]. For example, mobile applications use mostly geographic information to generate recommendations by considering the location of the user. For incorporating contextual information into the recommendation process, three methods of contextual pre-filtering, post-filtering and modeling have been used [17]. In the pre-filtering method, contextual information is used to select or construct the relevant dataset for rating prediction. Post-filtering ignores the contextual information initially. However, after doing the rating prediction using the entire data, it adjusts the result using contextual information for each user. Contextual modeling directly uses contextual information in the modeling technique as a part of the rating estimation.

Trust-aware (Community-based): this type of systems considers preferences of the users' friends for its recommendation. It is generally admitted that people tend to accept the recommendations from their friends rather than from similar anonymous people [27]. The day-to-day growing popularity of social networks raised the interest in community-based recommender systems also called social recommender systems. In general, the system obtains information about the social relations of the user and her friends' preferences and ratings to provide its recommendations.

Hybrid: these recommender systems combine the above-mentioned techniques to achieve higher performance. A hybrid recommender system, merging two techniques, tries to use the advantages of one to fix the disadvantages of the other. For example, collaborative filtering is not able to handle new-items without any ratings, while content-based approach does not face problems with new items since the recommendations are based on the items features which are easily available. There are different methods for hybridization [2]: □ Weighted hybrid – this hybrid technique uses a linear formula to

combine scores of each recommendation component. Thus, the components should have the ability to generate recommendation scores that are linearly combinable.

 Switching hybrid – in this approach, the system selects one recommendation technique among some candidates. There are different selection criteria, such as confidence value or external criteria, based on the experienced situation. Each component may have a different performance in different situations.

 Mixed hybrid – this approach relies on merging and presenting multiple ranked lists into one. Therefore, components should use the core algorithm to produce recommendation lists with ranks which can be merged into a single ranked list. The main issue is how to generate the new rank scores.

 Feature combination hybrid – there are two different recommendation components: contributing and actual recommender. The contributing component inserts features of one source to the source of the actual recommender. The actual recommender works with data modified by the other one.

 Feature augmentation hybrid – this hybrid is similar to the feature combination hybrids; however, it is more flexible and adds smaller dimensions since the contributor produces new features.

 Meta-level hybrid – it employs two components; the first one uses the model generated by the second as input. This is different from feature augmentation, which uses a learned model to generate features as input into the second algorithm. In this approach, however, the entire model is the input for the other one.

 Cascade hybrid – first, this method utilizes one recommendation component to produce the ranked list, and then the second component refines the recommendation list.

Pros & Cons of recommendation techniques:

Table 2 summarizes the current challenging issues organized in two classes: data and algorithms. The challenges are identified based on the features of the used techniques with respect to these two classes. In fact, based on the data and algorithms that each technique employs, the technique may face specific challenges with different scales. It is worth noticing that some of these challenges fall into both classes. Each technique has its own advantages and disadvantages and there is no single technique fitting with any system in any application. Depending on the application and sensitivity of the requirements, a more suitable technique can be selected. The challenges are listed as follows:

 User independence. Some systems, such as content-based and demographic recommenders, use profile or ratings of the user to build her own profile and it is independent of other users [28]. However, collaborative filtering and trust-aware systems need extensive user involvement and ratings from other users to find out the similarities among them.

 Transparency. Some of the recommender systems such as content-based recommenders are transparent [29]. They can explain how the system works by explicitly listing content features that caused the item occurrence. However, some other systems are unable to provide justifications for their recommendations. For instance, collaborative systems act as black boxes, and they do not provide much transparency.

 Cold start (New user). Recommender systems need to understand user preferences to generate accurate recommendations. The system reliability is generally lowered when it faces new users without ratings. This problem is even more sensitive for learning systems [30].

 Cold start (New item). Content based recommenders can recommend items not rated by any user, so they do not have problems with new items [31]. On the other hand, collaborative recommenders rely only on users' preferences and cannot recommend a new item which is not rated yet.

 Limited content analysis. Content-based techniques are limited to the number and type of features related to the items they recommend [32].

 Over specialization (creativity). This means that the system cannot provide something unexpected. A good content-based technique would not find anything novel since it recommends items similar to user profile [4].

 Knowledge acquisition. Most of the techniques can make recommendations according to user's rating, but knowledge-based techniques require obtaining additional knowledge before making the recommendations [33]. This is the most challenging issue which knowledge-based techniques are confronting.

 Data acquisition. The quality of recommendations is totally influenced by the relevancy and quality of collected data about users [32]. For explicit data, concerning the security and privacy issues, users are not willing to reveal their private and sensitive information such as demographic information. Implicit data is also difficult to obtain, for instance gathering contextual data in context-aware techniques which demands a high degree of interactivity and user involvement.

 Adaptive quality. Techniques using learning components can train themselves and improve the quality of their recommendations over time [34]. Techniques that work based on given rules, such as knowledge-based recommenders, have consistent performance and cannot improve themselves.

 Stability vs. plasticity (sensitivity to preference changes). Plasticity refers to the system capability to learn while reaching a stable state, while stability is the system ability to remain stable to disturbances or unimportant input data. Once a user profile has been learned and created in the system, it is hard to change its preferences [35]. For instance, a person with an interest in eating meat, who becomes a vegetarian will continuously receive steak recommendations (for example) for a while, until newer ratings report the change.

 Large historical data. People with a short history may not receive relevant recommendations as those with rich history profiles. Mainly, content-based and collaborative systems require large historical data to predict user rating. However, some systems including knowledge-based do not need any background data [33].

 Sparsity. Sparsity happens due to lack of information. In most of the applications, there are almost always many users that have rated only a few items. Some recommender system techniques such as collaborative-filtering generate users' neighborhoods using their profiles. When the user has rated just a few items, it is difficult to recognize her taste and she might be related to wrong neighborhoods [36].

 Shilling Attacks. As many users are relying on recommender systems to help them through their choices, inducing the system to change an item rating would be profitable to an interested party. Such efforts to influence the recommendations are called shilling attacks [37]. In such cases, anyone may give many positive ratings for their own materials and negative ones for their competitors.

	Open challenges	Content based	Collaborative Filtering	Context-aware	Demographic	Knowledge-based	Trust-aware	Remarks
Data	New item	√	×	√	×	√	×	Collaborative filtering and trust aware excel in applications where the set of items is stable or semi-stable.
	New user	×	×	×	×	√	×	New user in content based is more problematic than collaborative filtering.
	Large historical data	×	×	√	√	√	√	Collaborative filtering is more demanding for large historical data.
	Content limitation	×	√	√	√	√	√	Content-based is good for text-based recommendations but inefficient for unstructured items such as movies and music.
	Sparsity	√	×	√	√	√	×	Collaborative filtering is naturally sparse, yet there are some algorithms such as Matrix Factorization which alleviate the problem. Some of the content-based algorithms also have this problem, such as IF-TDF.
	User dependency	√	×	√	×	√	×	Hybrid techniques using content information can overcome bootstrapping problems.
	Data acquisition	√	√	×	×	√	√	Data acquisition is the first issue for all the techniques, but for context-aware and demographic techniques is more challenging.
Algorithms	Knowledge acquisition	√	√	√	√	×	√	Constraint-based methods are more challenged in knowledge acquisition.
	Adaptive quality	√	√	√	√	×	√	This problem is solved by machine learning-based algorithms in other techniques.
	Over specialization	×	√	√	×	√	√	The recommendations provided by demographic techniques are mainly too general.
	Stability vs. plasticity	×	×	×	×	√	√	This problem is more concerned with the methods using machine learning-based algorithms.
	Shilling attack	√	×	√	√	√	√	Hybrid-filtering and model-based methods cannot be biased easily.
	Transparency	√	×	×	√	×	√	Trust-aware is considered the most transparent technique.

√: Is not problematic ×: is problematic

Table 2. Open challenges for recommendation techniques

4.3) Development

In this section, we discuss the most popular and important algorithms that have been developed so far (see Figure 7). In general, content-based algorithms rely on the following steps:

1. Extracting the item attributes.
2. Comparing the attributes with the user preferences.
3. Providing recommendations by matching item characteristics with user preferences.

Vector-based representation and machine learning algorithms are mostly used for content-based recommender systems. Collaborative recommendation algorithms, as explained earlier in the first phase, have two general structures: 11

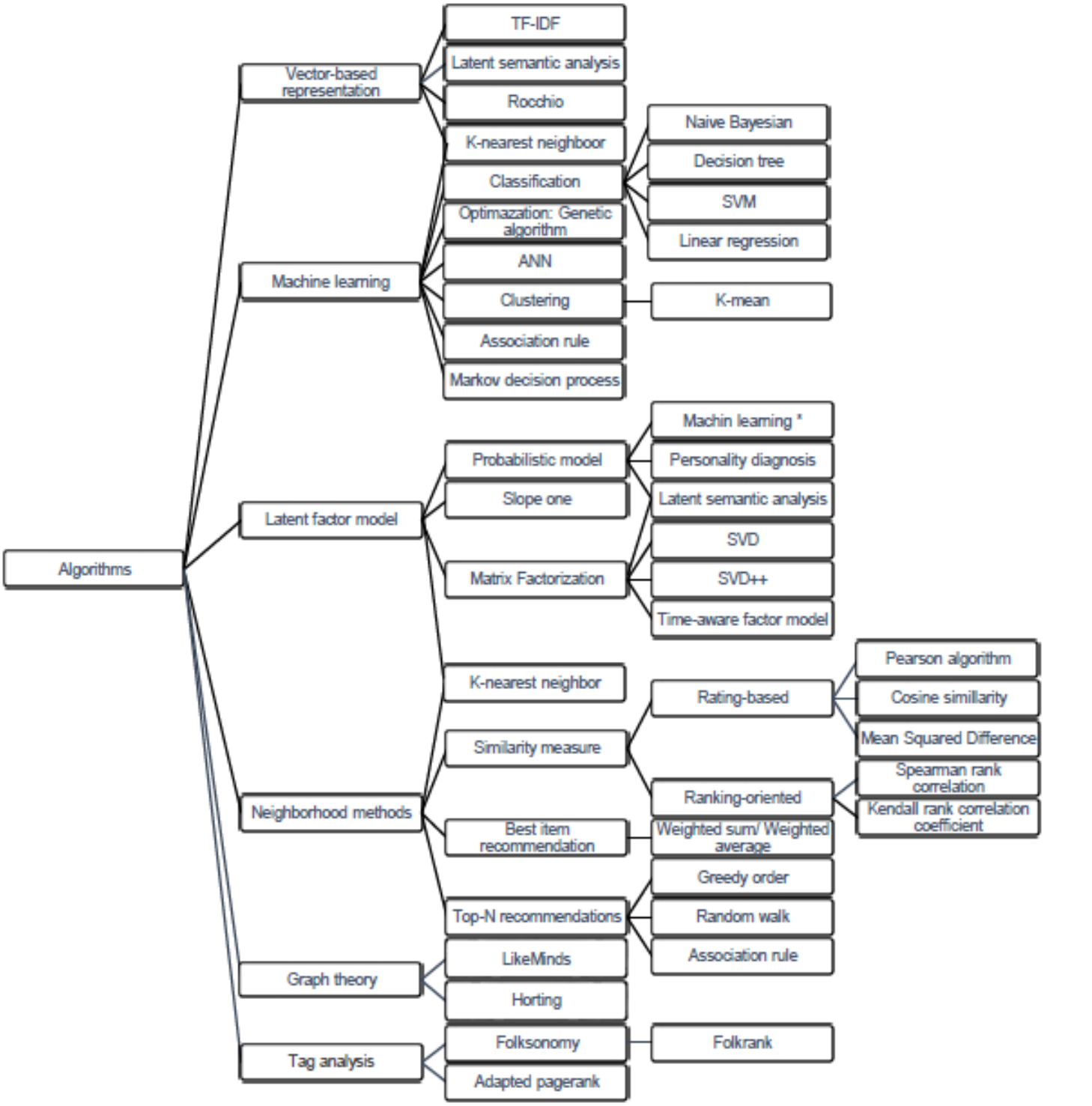
memory-based and model-based. Memory-based algorithms are essentially heuristics and predict ratings based on the entire previously collected rated items by the users, whereas model-based algorithms use the ratings collection to learn a model and make predictions for future ratings. We identify two major methods for implementing collaborative filtering with nearly the same categories of structure: neighborhood methods which are mostly memory-based and latent factor models which are model based [38]. This mapping is presented in Table 3. The rest of the recommendation techniques mainly use the same algorithms as the collaborative and content-based techniques.

Technique	*Structure*	*Algorithm*
Content-based		Vector-based representation
		Machin learning
Collaborative filtering	Memory-based	Neighborhood methods
	Model-based	Latent factor models

Table 3. Mapping major techniques into algorithms

Vector-based representation

A simple way to describe an item is to keep an explicit list of attributes or features of each item. When the user preferences are stated in terms of these features, the recommendation task would be matching them. However, it may not be appropriate to store “meta-information features”, which are additional knowledge rather than reflecting the content of the item. Therefore, content-based systems maintain a list of relevant keywords appeared within the item.



*Machine learning taxonomy is the same as above.

Figure 7. Taxonomy of recommender systems in the development phase

There are different ways of encoding the item in a keyword list. In a very first approach, a list of all words appearing in all items can be made and each item would be described by a Boolean vector (1: the word is appeared, 0: the word is not appeared). User profile can

have the same list of preferences in terms of vectors, so that for recommendation purposes, the overlap between item content and user interest should be measured. However, this method is not sufficient as it does not consider the weight of each word and also it tends to recommend items with longer descriptions, which have more words and more overlap [39]. The major algorithms using vector-based representation are: Term Frequency-Inverse Document Frequency (TF-IDF), latent semantic analysis, Rocchio and nearest neighbor. These algorithms are explained hereafter, while latent semantic analysis will be explained in the category of latent factor model under matrix factorization.

TF-IDF was proposed by Salton et al. to come up with the issue of traditional approaches [40]. It is designed to specify keyword weights in a content-based method. Content documents can be TF-IDF encoded as vectors in a multidimensional Euclidian space. The space dimensions correspond to the keywords appearing in the documents. The coordinates of a given document in each dimension are calculated as a product of two sub-measures: term frequency and inverse document frequency. Term frequency describes how often a certain term appears in a document. A keyword weight for a document is in direct proportion to the frequency of the keyword's occurrence in the document and in inverse proportion to the number of documents that the keyword appears in. A content profile can be represented as a vector of TF-IDF keyword weights. The similarity between two documents can be measured by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the frequency vectors. TF-IDF vectors are classically large and too sparse. To make them more concise while having relevant information from the vector, additional techniques can be utilized.

In information retrieval, the success of retrieving a document depends on the well-constructed documents, queries and keywords. There have been several proposed methods to help users refine their queries based on previous search results, called relevance feedback. The main principle is that users can rate the retrieved documents by considering what they need. In this context, *Rocchio's algorithm* is the standard relevance feedback algorithm operating in the vector space model [41]. The algorithm is based on the modification of an initial query by the mean of prototypes of relevant and non-relevant documents with different weights. This approach builds two document prototypes by taking the vector sum over all relevant and irrelevant documents. Theoretically speaking, this algorithm does not guarantee coverage and performance [42].

Nearest neighbor [43] algorithms store all the training data, called labeled items, in memory. When a new item or unlabeled item arrives, the algorithm compares it to all the stored items using a similarity function and identifies the k-nearest items to assign a class label to the new item. The similarity function used by the algorithm depends on the type of data. For structured data, Euclidean distance metric is usually deployed. When the vector space model is used, the cosine similarity measure would be suitable. Despite the simplicity of this algorithm, it can compete with complex learning algorithms. It is easy to implement, and can adapt quickly to the changes. Besides, it can make predictions with reasonable and reliable performance when a small number of ratings is available. However, the prediction accuracy is not very high compared to the best-known methods.

Machine learning

Deciding about suggesting an item to a user can be viewed as the problem of modeling a task and predicting if it is liked by the user. To this end, various standard (supervised)

machine learning techniques can be applied so that an 13 intelligent system can predict whether a user will be interested in an item or not. Supervised learning means that the algorithm relies on the existence of training data. The data might be collected through explicit feedback, or be obtained implicitly through observing the user. The rest of the section explains the main machine learning-based algorithms including classification algorithms, Genetic Algorithms (GA), Artificial Neural Networks (ANN), K-means, association rules, and Markov Decision Process (MDP).

Linear classifiers learn linear decision boundaries and separate instances in a multi-dimensional space. There is a large number of algorithms falling into this category, such as naïve Bayesian, decision trees and Support Vector Machines (SVM), and applied in content-based recommender systems successfully [42]. The training data of the classification learner is classified into the binary categories of items that the user "likes" or "dislikes". Decision trees are simple, understandable with reasonable performance for content-based models when only a small number of structured attributes is being considered. However, it tends to lead to poor performance on text classification as only few tests are possible [42]. SVM has a very good accuracy in text classification tasks even with noisy features. As reported in [34], there has been much research done on Naïve Bayesian classifier for modeling content-based recommendations. Bayesian classifiers are robust enough to isolate noisy and irrelevant features, and handle missing values by ignoring the instance during the probability estimations. Linear regression application in recommender systems can be summarized as a rating predictor based on neighbors rating and pattern identifier between the neighbors and active users or items [44]. Linear regression is more appropriate when the rating scale is discreetly continuous, for example between 0 to 10, or when the values are ordered in a clear fashion.

As argued in [7], many of the proposed recommender systems are based on bio-inspired methods such as *Genetic Algorithms (GA)* and *Artificial Neural Networks (ANN)*. Genetic algorithms are heuristic and based on evolutionary principles such as natural selection and survival of the fittest. They are mostly used for clustering, optimization and/or hybrid recommender systems [7]. Artificial neural networks are grounded on biological neurons' behavior. They try to simulate the way a brain processes information and learns to a certain degree. ANN model contains several interconnected nodes, each handling a specific domain of knowledge with several input networks. A node learns the relationships based on the inputs and operational feedback and model them to generate the desired output. The main advantage of ANN is its ability to perform non-linear classification tasks, and to operate even when a part of the network fails. They can be used to either construct a recommendation model or integrate the input from several recommendation modules [45].

Clustering is an unsupervised learning containing items assigned to groups, so that items belonging to the same group are the most similar to each other. *K-Means* algorithm is the de facto algorithm for clustering data. It partitions the dataset of items to several subsets forming a set of items, which are close to each other according to a given distance measure. It is simple and efficient but with few shortcomings and limitations: it needs prior knowledge to choose appropriate clusters, and it faces problems when dealing with clusters in different sizes and densities and when the outliers exist. Some studies applied K-Means to cluster the data and then to form the neighborhoods in KNN algorithm based on the clusters [23].

Association rule mining tries to find the rules that predict the occurrence of an item based on the occurrence of other items in a transaction. The main positive point of this classifier

is its expressiveness as it operates with data features without any transformation. This makes it easy to implement and interpret [46].However, like other classifiers such as decision trees, it is not really efficient to build a whole recommender system based on rules [34].In fact,rules are more appropriate when used to improve the recommendations by injecting some domain knowledge.

Markov Decision Process (MDP) algorithms view the recommendation process as a sequential optimization problem rather than a prediction problem [47]. MDPs model sequential stochastic decision problems and are applied when an agent is acting and affecting its surrounding environment. An MDP model consists of a four tuple: (*S, A, R, Pr*). *S* is a set of states, *A* represents actions, *R* is a real-valued reward function, and *Pr* denotes probability of transition between states given an action. An optimized solution is to maximize the function of its reward stream. Shani et al. defined *k* tuples of items as the states, and some null values referring to missing items [47]. Recommending an item corresponds to actions, and the utility of selling an item corresponds to a reward. The state coming after the recommendation is the user response to that recommendation, such as picking the recommended item or picking another item. The probability of buying an item depends on his current state, item and whether the item is recommended or not.

Latent factor model

Latent factor models aim to characterize both items and users using several factors to justify the ratings patterns. For instance, discovered factors for a movie recommender might measure genre, amount of action or age considerations for the movie; and for the users, how interested the user is on the movies with high score on the considered factors for a movie.

Latent factor models offer expressive ability to explain various aspects of the data and

provide results with higher accuracy than neighborhood methods' results in most of the cases [48]. However, concrete recommender systems use neighborhood methods due to its ability to provide recommendations based on just entered user feedback. Another reason is that they can furnish intuitive explanations for the reasoning behind the recommendations which enhances users affect [34, 49]. Latent factor model includes probabilistic models, slope one and matrix factorization which are introduced along with their related algorithms as follows.

Probabilistic model formulation of collaborative filtering was proposed in addition to employing probabilistic approach and combining similarity functions in other methods [50]. Probabilistic methods aim to build probabilistic models of user behavior patterns to predict future behavior. The main idea is to compute the probability that a user will pick an item, or the probability distribution over user ratings of the item.They can be deployed for ranking the recommendations as well.

Personality diagnosis is a probabilistic user model assuming that user ratings are a combination of their preferences and Gaussian noise [51]. The active user is assumingly generated by choosing one of the other users uniformly at random and adding Gaussian noise to her ratings. Given the active user known ratings, we can calculate the probability that she has the same "personality type" as other users, and the probability she will like the new items. Personality diagnosis can also be regarded as a clustering method with exactly one user per cluster. This approach is both model-based and memory-based.

The *slope one* algorithms are based on predictors of form $f(x) = x + b$, therefore, simpler than those used in the regression-based algorithms [52]. In the original slope one algorithm, the constant b is defined as the mean difference between each item and the

item to predict, computed among the users that have rated both items. Slope one performs well in sparse data and is computationally efficient [53].

Matrix factorization infers rating patterns to characterize items and users as vectors of factors. When a high correspondence between factors of item and user is observed, a recommendation is produced. Matrix factorization has become popular recently by promoting high predictive accuracy and scalability in dealing with large and multi-dimensional datasets [38]. Besides, it is more flexible in terms of modeling real world situations. In addition, it allows incorporation of supplementary information using implicit feedback. Matrix-Factorization is acknowledged by high accuracy in most of the research performed on collaborative filtering [54]. However, when computational efficiency becomes an important factor, slope-one could be a better option [53].

As discussed earlier, data sparsity is one of the major problems of collaborative filtering techniques and many methods have been proposed to alleviate this problem. Dimensionality reduction techniques, such as *Singular Value Decomposition (SVD)*, remove insignificant users or items to reduce the dimensionalities of the user-item matrix [34]. SVD converts a matrix B into three matrices. Given $m \times n$ matrix B, the SVD is defined as $SVD(B) = U \times S \times VT$. U and V are two orthogonal matrices of dimensions $m \times m$ *and* $n \times n$ respectively and S is a diagonal matrix of dimension $m \times n$ formed by the singular values of the rating matrix. U and V are called the left and the right singular vectors. SVD provides high accurate results, but it is highly expensive to update the data as it requires a re-computation for each new received rating. SVD is more suitable to be deployed in off-line settings when the known preferences do not change over time. A technique called folding-in was introduced to incorporate new data into an existing decomposition [55]. After a high amount of folding-in the decomposition misses its accuracy, and should

be updated. There are several algorithms for computing SVD, such as Lanczos' algorithm [56], the generalized Hebbian algorithm [57], and expectation maximization [58] which are very specialized and beyond the scope of this research to be described.

Latent Semantic Indexing (LSI), also known as Latent Semantic Analysis, is a well-known method in information retrieval for automatic indexing and searching of documents [59]. The approach benefits from the implicit structure (latent semantic) in the association of terms with documents and it is based on SVD, where user similarity is measured by representing users in a reduced space. The new matrices obtained represent latent attributes in the ratings, allowing finding relations among items and eliminating the problems caused by the sparsity of the matrix or anomalous ratings. LSI and SVD are typically combined [60].

SVD++ considers implicit feedback, leading to an increase in prediction accuracy [61]. Nevertheless, it is not limited to a special kind of implicit data. For the sake of simplicity, each user u is related with two item sets. One is $r(u)$ which contains all the items with available ratings from u. The other one is denoted by $N(u)$ and consists of all items which u provided an implicit preference for.

Matrix factorization can nicely model temporal effects to improve the accuracy when timing factors are considered, called *Time-aware factor model* [62]. Decomposition of ratings into distinct terms allows treating different temporal aspects separately. Especially, there are three factors which vary over time: user preferences, item biases (e.g., a movie may go in or out of popularity by the appearance of an actor) and user biases (e.g., a change in the user rating scale). On the other hand, the item characteristics are static in nature.

Neighborhood methods

Neighborhood methods compute the relationships between items (item-based) or, alternatively, between users (user-based). The item centered approach investigates the user's preferences for an item based on ratings of the neighborhood items by the same user. The user-oriented methods detect users with the same mind who can supplement each other's ratings [63].

There are two approaches in this method known as: similarity-based computation and top-N recommendation. They generally follow three steps:

1. Calculate the similarity or weight (referred as distance or correlation) between two users or two items;
2. Generate a prediction for the user by taking the weighted average of all the ratings of the user or item on a certain item or user.
3. In top-N recommendation, find the *K* most similar items or users after calculating the similarities, then gathering the neighbors to make the top-N most frequent items to recommend.

In what follows, we introduce the K-Nearest Neighbor algorithms (KNN) and the algorithms used for similarity measure, best item and top-N recommendations.

K-Nearest Neighbors (KNN) algorithms are the reference algorithms for collaborative filtering and work with similarity measures [63]. They have the advantage of simplicity and accuracy, but short fall in scalability and sparsity. KNN algorithms in item-item approach follow three tasks: 1- identify *q* items in the neighbor of each item in the database; 2- For item *i*, which is not rated by active user *u*, predict based on the ratings given by *u* from the *q* neighbors of *I*; and 3- select Top-N recommendations for the user. The first step might be conducted periodically to facilitate an accelerated recommendation regarding the user-user version.

A *similarity measure* identifies the similarity between pairs of users or pairs of items. The most used measures are as follows [5, 43]:

 Rating-based similarity measures are calculated by comparing rating values assigned to the items by different users.

 Pearson Correlation: measures the extent to which two variables linearly relate with each other. This method has problems in calculating high similarity between users with few common ratings. Setting a threshold on the number of co-rated items can alleviate this issue.

 Vector Cosine-Based Similarity: two users are modeled as two vectors in a multi-dimensional space and the similarity is evaluated by computing the cosine of the angle between them. It is worth mentioning that TF-IDF algorithms for content-based systems also use this metric to measure the similarity between vectors of TF-IDF weights. The difference is that this approach captures the similarity between vectors of the actual user ratings.

 Mean Squared Difference (MSD): it evaluates the similarity between two users as the inverse of the average squared difference between the ratings given by them on the same item. The negative point of this measure is that it cannot obtain negative correlations between preferences of the user or the appreciation of different items.

 Ranking-oriented similarity measures determine the similarity between users by their preferences over the items, which is reflected by their ranking of the items. Two approaches are distinguished:

 Spearman rank correlation: in this similarity function, the rated items by a user are ranked in a way that the highest rated item is the first rank and lower rated ones have

higher ranks. The computation is similar to Pearson correlation, except that the ranks are used instead of ratings.

 Kendall's τ correlation: it is like the Spearman rank correlation, but instead of using ranks themselves, only the relative ranks are used to calculate the correlation.

Many modifications have been proposed as extensions to the standard correlation-based and cosine-based measures to improve their performance. Some of them are weighted-majority prediction [64], default voting [65], inverse user frequency [28] and case amplification [66]. However, they mostly short fall whenever there are few user ratings since the similarity measure between two users is based on the intersection of the sets of items rated by both users [4].

Using the similarity measure, *Best item recommendation* aims to differentiate between levels of user similarity and estimates the best item by aggregation functions. To make a prediction for the active user, on a certain item, we can take a *weighted average* of all the ratings on that item. But the most common approach is to use *weighted sum* [4]. The similarity measure is essentially a distance measure which is used as a weight. The more similarity between users results in the more weight of rating occurring in the prediction of the new rating.

Top-N recommendation is a set of N top-ranked items of interest to a particular user which are mostly used in collaborative filtering and, also in some cases, in content-based techniques [67]. Top-N recommendation technique analyzes the user-item matrix to reveal relations among users or items and utilize them to generate the list of recommendations. User-based Top-N recommendation algorithms have limitations on scalability and online performance. Some models, such as association rule mining based

models [46], greedy order [68] and random walk [69] can be used to make Top-N recommendations.

- *Greedy order* algorithm searches through the possible rankings in an attempt to find the optimal ranking with maximum value. It generates a ranking from the highest position to the lowest position, by picking the item that presently has the highest potential and assigns a rank to the item which is equal to the number of remaining items, so that it will be ranked above all the other remaining items.
- *Random walk* differs from the greedy order algorithm by having the advantage of utilizing transitive relations among implicit preference functions. Instead of searching for a ranking directly as the greedy algorithm does, it attempts to define a Markov chain model with the transitional probability corresponding to a user preference function.
- *Association rule* that is a part of machine learning, is used in Top-N recommendation as a technique of identifying rule-like relationship patterns to detect groups of items that are favored together. For example, a rule may detect that "if a user likes both item 1 and item2, then the user will probably like item 5". Then it can generate a ranked list of recommended items based on the statistics about the co-occurrence of items in the sales transactions.

Neighborhood vs. latent factor model approaches:

The literature about recommendation techniques has shown that the model-based approaches are superior in terms of rating prediction accuracy [7, 43]. However, there is a rising awareness about insufficiency of predicting accuracy as the system effectiveness. Moreover, other factors, such as serendipity, are considered equally or more important. Model-based algorithms are suitable for capturing user characteristics and preferences with latent factors due to their learnable nature. On the other hand, neighborhood

algorithms can obtain local associations in data. For example, a movie recommender can recommend a movie different from what the users usually prefer or a movie which is not well known, if one of his neighbors has given it a good rating. Neighborhood methods are very simple and justifiable for what they predict, meanwhile they enjoy efficiency without expensive training [49]. However, the recommendation production is more expensive to re-compute upon the arrival of new ratings. This can be overcome by an off-line pre-computation. Neighborhood methods are also more stable and less affected by adding users, items and ratings [63].

As a matter of fact, the appropriate algorithm for a certain application should be selected by considering a combination of the characteristics of the target domain and the context of use, the requirements of computational performance of the application, and the user's requirements.

Graph theory

In today's social networks, web users reveal their relations and connections among other users, where the posted images and videos are shared within their trusted network. Users also provide more information on their demographic characteristics and preferences willingly. Standard collaborative filtering algorithms are not able to find sufficient similar neighbors in sparse datasets. Thus, trusted social relationships of users emerged as an alternative improvement for recommender systems [70]. To model trust networks, some methods have been used such as machine learning, semantic models, fuzzy models and graph theory. Trust network can be a dedicated graph in which users are nodes and edges are trust relationships. The edges can also be weighted to show how strong the trust is.

In the context of recommender systems, graph representation of ratings is called hammock. A hammock of width *R* links two users sharing a minimum of *R* ratings. To connect two users, a sequence of hammocks can be used. A recommendation is generated by comparing user feedback and predicted ratings from hammock paths. Different algorithms use hammocks in different ways to make recommendations, and the most important ones are LikeMinds and Horting [71]:

- *LikeMinds*: it uses a single hammock to signify the importance of *R* shared ratings in terms of recommendation quality. To predict the rating of item *i* for user *u*, LikeMinds computes the agreement scalar between *u* and every other user who has rated *i*. The algorithm utilizes the ratings of the users with the highest agreement scalar to generate the recommendation. The ultimate goal of LikeMinds is to offer fast and accurate real-time personalization.
- *Horting*: this algorithm was developed by IBM research to overcome the sparsity issue by not requiring a direct link between the user and the item. Horting exploits explicit hammock paths of varying length to produce its recommendations. It uses a transformation technique similar to the one used by LikeMinds to perform predictions based on the other users' ratings. However, unlike LikeMinds, it accommodates not only the pairs of users with highest agreement scaler, but also the ones with similar or even opposite ratings, as well as a combination of them.

Tag analysis

Tagging allows users to annotate the content with any kind of label in the web environment. Tags can be applied to any kind of items, even users. This plays a key role in sharing content across the social networks. The following is a list of the most popular examples of tagging systems:

 Folksonomy: free annotations on the web formed folksonomies. The notion of folksonomies is taken from folk-generated taxonomies which perform tagging in a horizontal and inclusive way rather than in a hierarchical way. This assures users to get all relevant items in one query. A folksonomy is a tuple *F = (U, T, R, Y)*, where *U* is a user, *T* represents the tag and *R* is a resource. *Y* defines the relation between them. A folksonomy exploits the information of how items are tagged by the community for predicting interesting items to a user. A number of algorithms (e.g. Affinity Propagation) have been proposed to obtain folksonomies [72]. One of the issues of folksonomies is that users' ways of annotating are different; therefore, fuzziness approaches have been introduced in the tags.

 Folkrank: it was developed with the idea that a resource is important if it is tagged by influential users [73]. Folkrank algorithm is inspired by the PageRank algorithm and is a graph-based search and ranking method for folksonomies.

 Adapted PageRank: the PageRank algorithm is graph-based and emphasizes on the fact that a web page is important if it is linked with many pages which are important themselves. Due to the different nature of folksonomy compared to the web graph, PageRank cannot be applied directly into folksonomies [74]. This algorithm has proven to be one of the top performers in tag recommenders. However, it imposes high computational costs [75].

4.4) Evaluation

Recommender system evaluation can be conducted either online or offline. Offline evaluation is the simplest approach since it does not need to interact with real users. For a real-world recommender system, it is desired to be tested online to investigate the system influence on user behavior. However, this experiment is expensive when the

performance of many algorithms should be compared, and it is difficult to completely understand the relation between the user and the system properties. There is an alternative to these two evaluation methods which is called user experience (user study) [76]. In this case, a small number of users are asked to use the system in a moderated environment and to report their experience with the system.

A complete list of measures for evaluating recommender systems performance, which is shown in Figure 8, has been proposed in [5, 77]. However, many of these measures

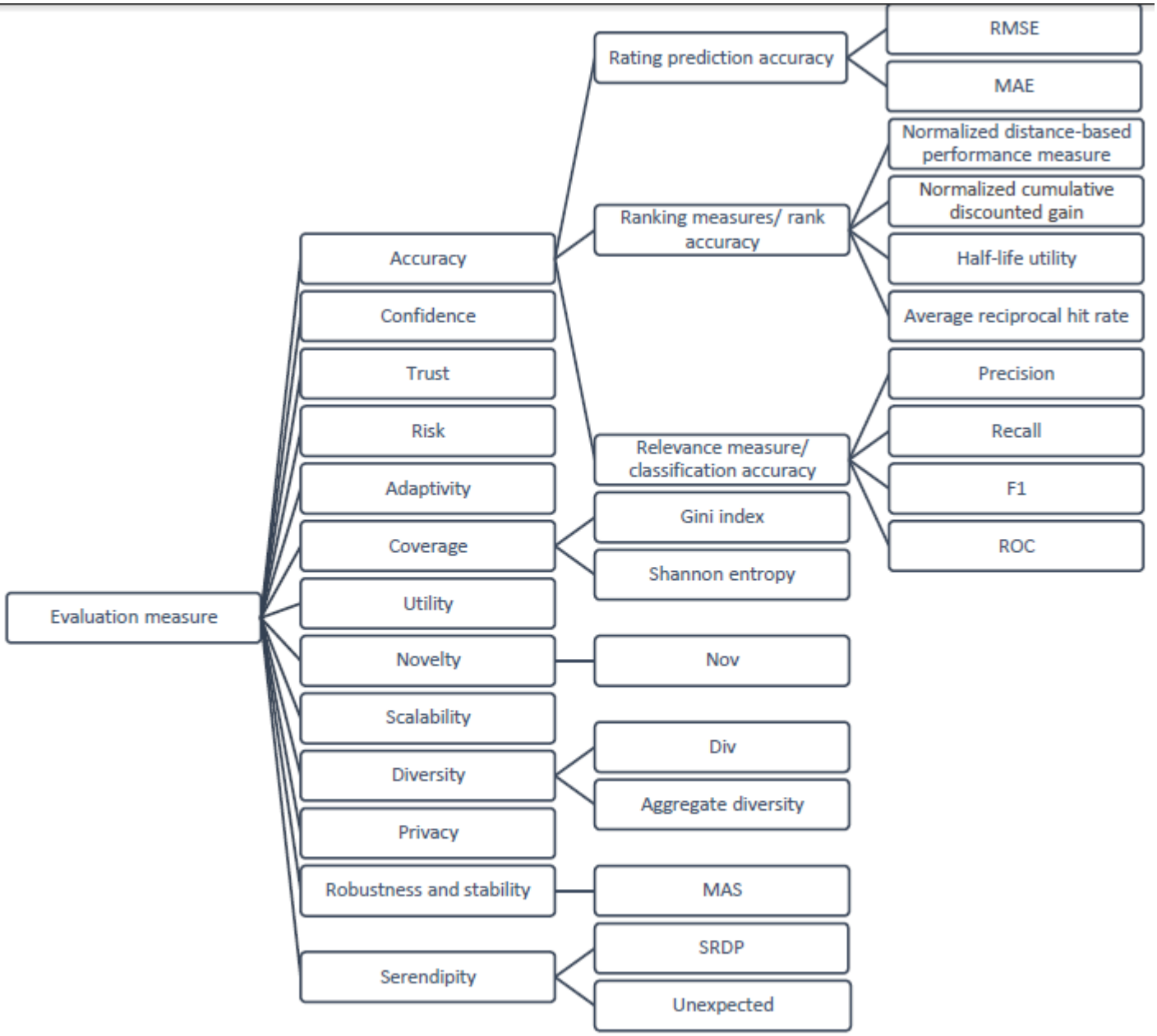


Figure 8. Taxonomy of recommender systems in the evaluation phase

remained theoretic and inexperienced. Table 4 provides a brief introduction about these measures and more details can be found in [5, 77]. In this section, the most recent progress made in this area is discussed.

Measure	*Definition*	*Challenge*
Accuracy	It measures the difference between the real rating/ranking and the rating/ranking predicted by the system	Rating measures cannot distinguish lower rating from medium rating. The relevancy measure needs binary user preferences.
Confidence (Reliability)	This measure is defined as the reliability of recommendation and the system's trust in its recommendations or predictions and can be reported by the confidence score of the system.	The system' algorithms should agree on the confidence method.
Trust	Trust refers to users trust in the recommendations provided by the system.	It must be measured in online studies with direct feedback from the users.
Risk	It regards the applications of recommender systems that are associated with a potential risk such as recommendations of stocks in the market.	Expected utility and utility variance should be identified.
Adaptivity	Recommender systems may operate when the items change rapidly, or when user's interest over items may shift such as news recommenders.	The speed of the trends' change affects the recommendation accuracy.
Coverage	It is the percentage of items that the employed algorithm could produce recommendations for.	Cold start is a sub-problem of coverage.
Utility	Utility is defined as the value which an owner of a recommender system or a user may gain.	It is difficult to model user utility and aggregate utilities from all users to compute the system score.
Novelty	Novelty designates the difference degrees between the recommended items and known items by the user.	Irrelevant new items should be avoided.
Scalability	Recommender systems are developed to guide users to navigate large collections of items. The main goal is to scale up to the real datasets.	It trades accuracy and coverage to speed up for huge datasets.
Diversity	Diversity denotes the degree of differentiation among the recommended items.	Diversity sacrifices the prediction accuracy.
Privacy	Privacy concerns users' private information.	It has an exchange with most of the other measures.
Robustness and stability	Robustness measures the system performance before and after a shilling attack, but stability looks at the systems' predictions for possible shifts in the attacked items.	It is hard to execute an attack on a real system as an online experiment.
Serendipity	It is a degree of how interesting and surprising a recommendation is. A serendipitous item should not be expected by the user nor discovered yet, while staying useful and interesting.	Extra data is crucial.

Table 4. Definition of evaluation measures

Despite the comprehensive evaluation measures proposed so far, there is no standard technique and functional algorithm to evaluate most of them. The measure which has

been most experimented with on recommender systems performance is accuracy. Another important measure is reliability as proposed by A. Hernando et al. [78]. Although it is a common metric, it is limited to recommender systems using KNN algorithms. The definition of reliability on the prediction is based on two numeric factors: one measures the similarity of the neighbors used for making the prediction, and the other one measures the degree of disagreement between the habits of neighbors in rating the items.

A variety of metrics has been used to measure novelty and diversity of a recommender system [79, 80]. Most of the methods proposed to evaluate diversity use item-item similarity based on the item content to form item neighborhoods, then measures the min, max, sum or average distance between pairs of items. Another way is to measure the value of inserting a new item to the list of recommendation, as done by Ziegler et al. [81]. Adomavicius and Kwon used the total number of different items recommended across all users as an aggregate diversity measure to measure the recommendation algorithm performance based on the Top-N recommended items lists (*Lu*) that the system provides to its users (*u*) [82].

$$Aggregate\ Diversity = \left| \bigcup_{u \in U} L_u(N) \right| \qquad \text{Eq. 1}$$

Hurley and Zhang suggested the following novelty (*Nov*) and diversity (*Div*) measures for Top-N recommendations in collaborative filtering item-based methods [67]:

$$Nov = \frac{1}{\#L_u - 1} \sum_{i \in L} [1 - sim(i,j)] \qquad \text{Eq. 2}$$

$$Div = \frac{1}{\#L_u(\#L_u - 1)} \sum_{i \in L} \sum_{j \in L} [1 - sim(i,j)] \qquad \text{Eq. 3}$$

where *u* is the set of users, *i* and *j* are the items, and $\#Lu$ indicates the number of recommended items. The function of *sim* measures the similarity of item *i* to item *j* that refers to the similarity measures explained in Section 4.3 under the category of neighborhood methods. In fact, $1-sim(i,j)$ denotes the distance or dissimilarity between the two items. In another study towards novelty and diversity, Vargas and Castells developed a formal framework to define these two measures in order to unify the current state of the art measures [83]. They proposed two novel features of "sensitivity" and "relevance awareness" in novelty and diversity measurement rank through a probabilistic recommendation browsing model. Occasionally, the defined metrics of precision and recall for accuracy evaluation have been used to assess other measures such as novelty and robustness [84].

To the best of our knowledge, experimental studies on serendipity are rare. Ge et al. tried to measure serendipity using a benchmark model generating expected recommendations [85]. When the recommendation provided by the recommender system does not belong to the benchmark model, it can be concluded as an unexpected item (*UNEXP*). The authors also argued that not all the unexpected recommendations are useful (*USEFUL*). Considering this point, they defined a new serendipity measure called SRDP: (Eq4)

$$SRDP = \frac{|UNEXP \cap USEFUL|}{|N|}$$

Adamopoulos and Tuzhilin revised SRDP measure and proposed a new measure by defining expectedness as the mean ratio of those items in the consideration set of a user (Eu) as well as in the generated recommendation list (Lu) [86]. The value of serendipity (UNEXPECTED) is computed based on the mean ratio of those items that are not included

in the set of expected items for the user but are included in the generated recommendation lists: (Eq5)

$$UNEXPECTED = \sum_{u} \frac{|(L_u/E_u) \cap USEFUL_u|}{|N|}$$

Based on the fact that a stable recommender system does not change its prediction strongly over a short period, a quality measure named Mean Absolute Shift (MAS) was proposed by Adomavicius et al. [87]. MAS is defined by a set of predictions of all unknown ratings *P1*. For a period of time, users would rate a subset *S* of these unknown ratings and the recommender system can start make new predictions of *P2*: (Eq6)

$$MAS = \frac{1}{|P_2|} \sum |P_2(u,i) - P_1(u,i)|$$

4.5) Application

Thanks to the development of efficient and advanced recommendation techniques and algorithms, more industries and businesses have employed recommender systems. Park et al. revealed that considering the age of big data, applications analysis is the main focus of current recommender systems studies [9]. Lu et al. defined eight application domains for recommender systems [8]. As shown by the statistics presented in Section 3, there is a widespread adoption of recommender systems in social media and health-related fields. We adopted Lu's list of applications and refined it in our new classification as illustrated in Figure 9. We considered three main areas, namely entertainment, education, and service. The applications of recommender systems in each of these domains are described briefly. To show how recommender systems were practiced with in the academic and commercial world, Table 5 provides one sample for each application through the explained system lifecycles.

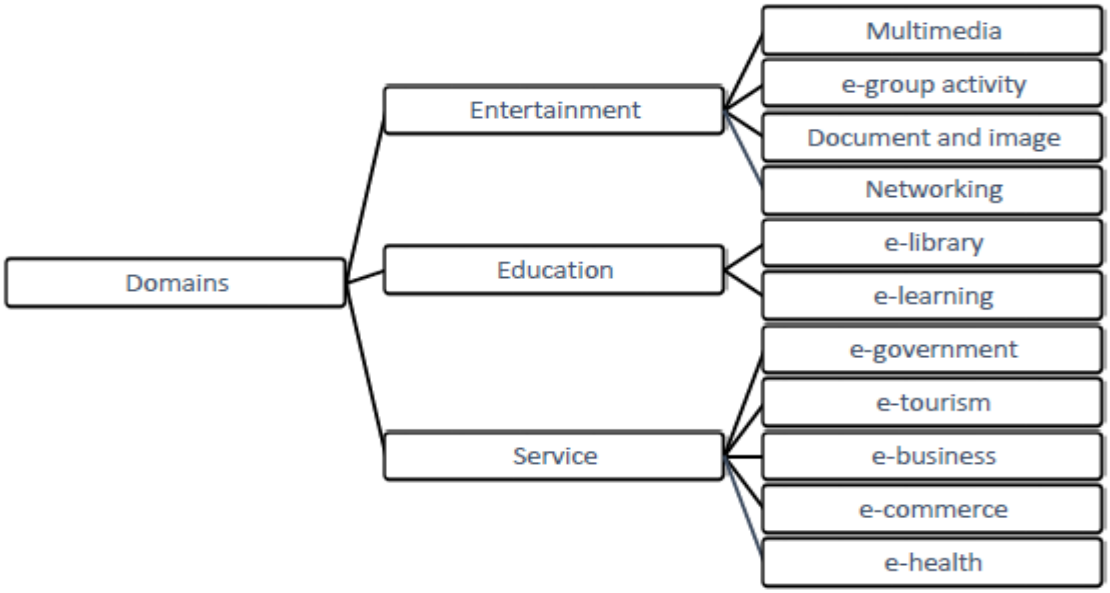


Figure 9. Taxonomy of recommender systems in the application phase

Reference	*Approach*	*Technique and algorithm*	*Evaluation*	*Application*
CofoSIM [88]	– Multi-criteria – Implicit data	Collaborative filtering (KNN)	Accuracy (MAE), precision-recall, F1	Multimedia (music)
News Dude [89]	– Individual user – Implicit and explicit data	Collaborative filtering (Bayesian algorithms)	Accuracy (Precision-recall, F1)	Document
GRec_OC [90]	– Item and user – Group users	Hybrid of content-based and collaborative filtering (KNN)	Accuracy (Precision)	E-group
Friendbook [91]	– Individual user – Context and user – Personalized – Implicit and explicit data	Context-aware (Graph theory, Latent Dirichlet Allocation algorithm), semantic-based probabilistic model	Accuracy (Precision-recall), scalability	Networking
Smart learning recommender [92]	– Individual user – Implicit and explicit data – Personalized	Hybrid of content-based and collaborative (clustering and KNN), and knowledge-based	Simulation of user experiment	E-learning
ACM [93]	– Implicit data – Item – Non-personalized	Hybrid of content-based and collaborative filtering (Fuzzy classification)	Not specified	E-library
Google Scholar/ JSTOR [94]	– Non-personalized – Item – Implicit data	Content-based (ranked by PageRank algorithm)		
CiteSeer [95]	– Item – Explicit and implicit data	Content-based (weighted sum of TF/IDF)		
BizSeeker [96]	– Item – Explicit and explicit data	Collaborative filtering (semantic similarity, weighted sum)	Accuracy (MAE), coverage	E-government
Finance advisory [97]	– Centralized – Implicit data	Knowledge-based (case base reasoning)	Accuracy Diversity	E-business
Cosmetic recommender [98]	– Item and user – Individual user – Multi-criteria	Hybrid of demographic, content-based, and collaborative filtering (clustering and association rule mining)	Confidence	E-commerce
3D-GIS recommender [99]	– Context, user, item – Centralized	Hybrid of context-aware, knowledge-based, collaborative filtering	User experiment	E-tourism
Web-based health recommender [100]	– Explicit data	Knowledge-based (rough set)	User experiment	E-health

Table 5. Applied recommender systems in academic and commercial settings

Entertainment

Recent years have seen an increasing interest in the application of recommender systems in multimedia including movies, music, TV programs and online materials. Movie and music recommender systems mainly use collaborative filtering based on users' ratings. An example of such systems is CoFoSIM that makes use of implicit ratings in the context of mobile music market [88]. TV program recommender systems rely heavily on content-based techniques. However, since content information is described by features, they may also adopt collaborative filtering with the possibility of rating, specifically with new smart TVs [101]. Online documents, images, web pages, emails and newspapers can also benefit from recommender technologies. In most cases, a list of keywords is extracted from historical data or search engines/URLs as textual content. The recommendations are provided based on the analysis of these keywords using mainly probabilistic models [89]. From the social perspective, researchers have designed group recommender systems to combine the individual expectations of users in two ways: offline, meaning the group is already formed, and online, which means the system should make the grouping. In fact, the Internet has changed the way people socialize and so user profiles with known social links can nourish the input of recommendation techniques for recommending friends or social communities. Trust links are mostly computed based on the social network analysis using graph theories. Social networking applications rely primarily on trust and context-aware techniques to generate efficient recommendations [91].

Education

E-learning recommender systems aim to help learners choose the courses, subjects, materials and their learning activities such as study group discussions. Knowledge-based techniques play a very important role in developing such recommender systems when

there is no sufficient historical data. Otherwise, simulated users and ratings have to be investigated [92]. Application of recommender systems in digital libraries assists users in finding and selecting information and knowledge sources. The proposed recommenders in research mostly utilized hybrid techniques with fuzzy models to take advantage of different techniques and manage information of linguistic labels [8]. However, in real world digital libraries, deployed systems are practiced with less complex techniques [102].

Service

To support citizens and businesses to access personalized public services, such as finding a proper business partner and getting appropriate event recommendations, the government can adopt recommender systems. The most deployed approach in this context is item-based collaborative filtering with semantic similarity, as done in [96]. Another example of service is provided by e-shopping systems where rating is a common feature that can be used for recommendations. Many large commercial websites, such as eBay or Amazon, have already used recommender systems to suggest relevant products to different customers. These systems recommend products based on top sellers, costumer demographic profiles and past purchase behavior of returning costumers. There are several recommender systems with a variety of techniques in individual business applications, such as beauty and make-up [103], property rental [104], stock market [105] and Finance advisory [97]. Moreover, tourism recommender systems create substantial opportunities for tourists to get advice, for instance on their mobile devices, for a variety of attractions, destinations, tour plans, transportation, restaurants and accommodations. Among these attractions, restaurant recommender systems are of high interest. In health recommender systems, the items of interest for users are a piece of

non-confidential medical information that is scientifically proven. These systems mainly use knowledge-based techniques by leveraging the expressiveness of ontologies [106].

5. New Insights and Potential Advancements in Recommender Systems

Helping users handle the issue of information overload was perceived to be the original task of search engines or information retrieval systems [107], but what make recommender systems distinct from search engines are the criteria of being "*personalized, interesting and useful*". In fact, when a user is using a search engine, she knows what she is looking for, and makes the query accordingly. In contrast, recommender systems operate when the user does not know what she wants or likes, but the system knows the user' tastes; finds items that she prefers. In fact, a search engine requires the user to formalize a query to receive the information, while recommender systems notify the user with possible useful information without the need of an explicit query. Thus, search engines and recommender systems come up with a different scope towards the same goal, which makes them complementary. Nowadays search engines are equipped with recommender systems.

What makes a recommendation more *interesting* and *useful* is the factor of "*intelligence"*. Intelligence is the key core of personalization to understand the user's preferences, predict user's unknown favorites, and at the end provide recommendations beyond a simple search by matching the query and the content. Recommender systems research has incorporated a wide variety of Artificial Intelligence (AI) techniques including machine learning, data mining, user modeling, case-based reasoning, and constraint satisfaction, among others. The idea of having an intelligent system, which can think and learn like a human, led into more humanized techniques called Computational Intelligence (CI). CI is a branch of AI that explores the adaptive mechanisms to enable intelligent actions within

the compound and changing environments [108]. Thus, CI exhibits those AI paradigms which can discover and infer new information, to learn and adapt to different situations, and to generalize and make associations. In general, five techniques have been defined for CI [109]: 1) Fuzzy Sets (FS), 2) Artificial Neural Networks (ANN), 3) Evolutionary Computing (EC), 4) Swarm Intelligence (SI), and 5) Artificial Immune Systems (AIS). CI techniques initially originate from the human biological system. ANN simulate the biological neural system, EC is a replicate of natural evolution (e.g., genetic and behavioral evolutions), SI represents social behavior (e.g., organisms living in swarm or colony), AIS model the immune system (learning to produce the right antibodies to fight over each antigen), and finally, FS study how different organisms interact within their environment. Earlier, we explained how ANN-based technique has been employed as a recommendation algorithm. The other CI techniques are also receiving prominent attention from researchers working in recommender systems since they have significant potentials to make recommender systems more robust, effective, personalized and even context-aware [110]. Regardless of the fact that each individual technique has been successfully applied, the current trend is to use a hybrid solution as there is no superior one in every situation.

Figure 10 illustrates our layered framework for recommender systems containing market strategy, data, recommender core, interaction, security, and evaluation layers. The relevant existing techniques which can be applied in different layers of the system are identified (own as grey boxes). Moreover, we propose some new techniques which have high potential to be employed in this area and can overcome some of the challenges summarized in Table 2 (e.g., *cold start*, *sparsity* and *scalability*) that recommender systems are facing but have not investigated yet in the literature. The broad and diverse

practical applications of recommender systems can inspire researchers to explore novel and innovative solutions that will expose these systems into interesting but even more challenging areas. In the following, we introduce each layer and focus on advanced insights.

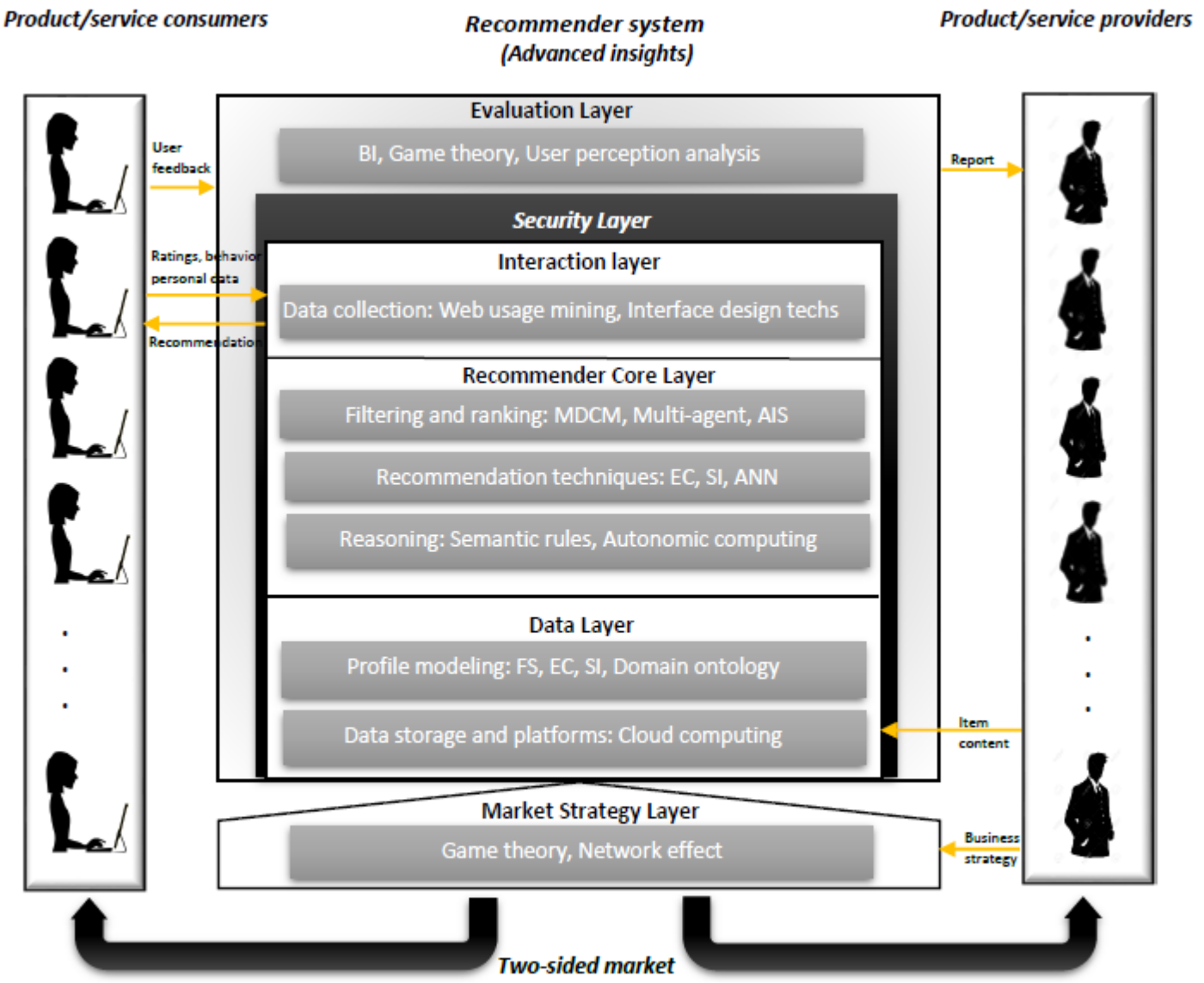


Figure 10. New and existing advanced insights into different layers of recommender systems

Market Strategy Layer

The market strategy is among the key elements of future recommender systems, where the system itself and market participants are derived by business and market strategies

[6, 111]. In today's market where values are created by users' networks, product and service providers can no longer compete by simply comparing features and prices to gain the competitive advantage. Recommender systems can be empowered with theoretical foundations such as network effect, two-sided market and game theory. According to the theory of network effect, a product or service becomes more valuable to its consumers and more profitable for its providers as more users use that same product or service. Beyond trust-aware and community-based recommender systems that work in socially-trusted networks, other types of recommender systems, such as collaborative or context-based, can benefit from this theory. In fact, designing recommender systems as online two-sided platforms will provide a rich source of data (ratings, items, users, etc.) and will open the door to innovations in terms of dealing with data as commodity. Moreover, as stated by the two-sided market theory, it is enough to induce either the consumers or providers to use the platform of recommender systems, so that the other part becomes motivated to join the platform [112]. The main challenge is to determine the type and amount of incentives to provide to each side (users and providers) so that the other side will follow. Thus, the research questions that need to be addressed, and which will open interesting research directions are 1) which part to be subsidized and which part to be charged; and 2) which pricing schemas and mechanisms to be used. In this context, laying theoretical and algorithmic foundations for novel subsidizing and pricing mechanisms of recommender systems is a highly appealing objective. This can be accompanied by studying users' perception and unfolding their sensitivity towards price and quality. The situation gets more complicated when multiple recommender systems are to be considered. Recommender system owners should decide whether to compete or

collaborate with other recommender systems. Thus, deep analysis of coopetition strategies will need to be theoretically and experimentally conducted.

From the game-theoretical perspective, there can be serious restrictions to achieve an equilibrium strategy among item/service providers, item/service consumers, or both. First, each participant only has the knowledge of her own ratings and recommendations, while others' ratings cannot be observed, which makes the game information-incomplete. Moreover, the recommendation algorithm adopted by the system is unknown to participants. Second, since there is no perception of users' expectation for recommendation quality, it would be hard to determine the optimum gain of the users in a game. The effects of ratings on providers' income and the analysis of the uncertainty are yet to be investigated.

To overcome the *sparsity* and *cold start* problems of "new user" and "new item" summarized in Table 2, researchers focused on developing more reliable prediction models for situations in which only a few item ratings exist. Most of these approaches depend on adjusting the algorithm that determines a recommendation. However, it is acknowledged that rating incurs some costs (time and privacy cost) for the user. Consequently, users may not rate or only rate few items that they have experienced. In this context, marketing and strategy-oriented approaches using game theory and two-sided market theory with a motivation and rewarding system that can incentivize consumers to participate and to report true ratings are potential alternatives to be explored.

Data Layer

The data layer oversees the processing of the input data obtained from the interaction layer. Contextual information, items and user profiles gather explicit and implicit data for

inference of similarities or modeling. Referring to *data and knowledge acquisition* challenges in Table 2, user preferences can be vague and gradualness, which makes them difficult to analyze. FS can overcome this problem by providing flexible methods to manage non-stochastic uncertainty. FS provide the possibility of having fuzzy criterion where evaluation of items is represented in relation to its possibility to belong in one of the intervals of a qualitative or descriptive evaluation scale. For example, Cornelis et al. [113] presented user preferences as two fuzzy sets of positive and negative feelings; from user set to item set. Content-based is developed using a fuzzy relation within an item set to compute item similarity. Collaborative filtering is generated based on the user preferences by fuzzy relations between the items. Composing these fuzzy relations provides the final list of recommendations containing positive and negative preferences. Evolutionary techniques, such as GA, are also able to extract implicit information from user logs, and can be combined with fuzzy techniques to include vagueness in decision making. This hybrid approach may allow more accurate and flexible modeling of user preferences. SI techniques can be another alternative to model users with relative accuracy and simplicity [110].

Item and user profiles can be stored in ontological repositories. More advanced techniques are being applied with the advancement of the web and Internet facilities. Web 3.0, or semantic web, opens new opportunities by providing semantic information about users or items and improves classical filtering methods [114]. Filtering techniques and similarity measurement can be improved by incorporating the semantics using domain ontologies and the set of concepts associated to item and user profiles. They can alleviate several issues discussed in Table 2, such as *sparsity, large historical data* and *cold start*.

In the era of “big data”, recommender systems have to deal with a huge volume of data and a large scale of computational tasks and costs. Cloud computing and distributed platforms can provide the opportunity to overcome these issues. The dynamic structure of cloud and its elasticity make cloud platforms a convenient host for providing the future recommendation services.

Recommender core layer

This layer supports the main activities of the system that contains analysis and reasoning, recommendation techniques, and filtering and ranking algorithms. In concrete and real settings, humans require some data to provide recommendations to one another. However, having the data is not the end of the story. The data should be processed and reasoned to further obtain information and knowledge. Similarly, to make a recommendation, the recommender’s brain should infer knowledge from the data, make a logical link between the data with prior knowledge and generate a recommendation thought to be suitable and helpful. In this process, ontology-based repositories discussed in the data layer, and inferred semantic rules can be helpful in data analysis and *knowledge acquisition* (see Table 2). As an example, Maidel et al. [115] used two ontological profiles of users and items to develop a similarity function between user interests and items. Another method of utilizing semantics is to exploit semantic information of items for computing the similarity between them [116]. Then these similarities can be combined with the similarities revealed from past user ratings to predict future user ratings. However, all these methods need semantic information which is hard to be acquired. Researchers are trying to develop systems which are able to generate semantics with the least human intervention [117].

ANN have been trained to learn users behaviors with data obtained from web usage mining in the interaction layer, and then group users into different clusters that possess similar preferences [118]. This makes ANN a suitable technique to address the challenge of *adaptive quality* (see Table 2). The weights and fitness functions derived from ANN training can be optimized using GA to obtain more accurate classification rules. GA itself can be used for search optimization. GA can bring randomness in the content filtering instead of strictly adhering to user profiles [119]. This can eliminate the *content limitation* challenge (see Table 2) and bring serendipity into recommendations.

Deep learning is a promising alternative to the conventional neural networks [120]. In recommender systems, there are many entities and properties assigned to the items and users, finding the proper feature (feature extraction and feature selection) is vital to improve the quality of classification and clustering methods. In recommender systems that involve users' behaviors, the most effective features can be a complex combination of the system properties, which are hard to be extracted and modeled by ANN. Deep learning methods are superior in effective feature learning, especially when there is no known effective feature. Deep neural networks learn the effective features representation automatically, and there is no need to specify and hardcode the features in the design level. The outcome of deep learning is usually surprisingly unexpected, specifically when there is no supervised class. Modeling and designing recommender systems as online two-sided platforms will benefit from deep learning to effectively learn and predict the strategies of the users and providers enabled by the amount of data provided by the platforms.

Providing a suitable solution for the problem of *stability vs. plasticity* (see Table 2), the SI algorithms, such as PSO, can attain feature weights for the user, and therefore, help

adapt the matching function to the user's specific tastes. Personalized recommendations based on individual user preferences or collaborative filtering data have also been explored using PSO. This was done by building up profiles of users and then using an algorithm to find profiles similar to the current user using supervised learning [110].

For ranking and visualization prioritizing purposes, engaging MDCM modeling may allow us to explore alternative recommendation forms and avoid *over specialization* problem (see Table 2). For example, rather than recommending a list of items with top-N utility values, an item list including the best performance for specific criteria can be suggested. However, it requires using more complex modeling methodologies with multi-objective optimization. In multi-objective optimization, more advanced algorithms that can handle possible conflicts in objective functions are yet to be put forward.

Most of the recommender systems are developed on centralized architectures. However, modern recommender systems are compelled to operate in small-scale and mobile devices within peer-to-peer environments, with less computing and storage requirements. The multi-agent approach in recommender system design is a flexible method for dynamic adaption with user requirements. For instance, D. Rosaci and G. M. Sarn developed an e-commerce recommender system using four intelligent agents in a distributed fashion [121]. They considered a device agent exploiting the user device, and a customer agent representing user profile. Further, a hybrid of collaborative and content-based filtering was developed using two agents: seller agent and counsellor agent. The scalable architecture of the system allowed it to operate within large communities, and upon various devices, sometimes with limited resources. A combination of intelligent agents and AIS techniques seems to bring us closer to an adaptive and scalable recommender system.

With the advancement of recommender systems and expansion of the dynamic requirements for multi-channel, multi-criteria and adaptive systems, there will be a need to explore more novel modeling options. An interesting research direction towards the design of adaptive recommender systems is to advocate self-managed systems that can autonomously choose the appropriate recommendation algorithm based on the device, situation, and properties of the application. An autonomous recommender system would be capable of self-directed learning and adapting its behavior to suit its context of use where rapid scalability and adaptability with underlying platforms are required across a large and diverse community. Thus, autonomous recommender systems can effectively 1) address the issue of *stability vs. plasticity* by handling sensitivity against preference changes; and 2) shield against *shilling attacks* by detecting unexpected behavior changes (see Table 2).

Interaction layer

In the interaction layer, the system's user interface is designed and implemented. This layer interacts with the core recommendation layer, which decides where to store and how to handle the data. As discussed earlier, the design of user interface varies depending on the device and application context. It contains visual components, interacting with the user, and non-visual components sending and receiving commands from the other layers.

Web usage mining or data mining techniques are being used as implicit methods to collect inputs for user modeling [122]. When a user interacts with the web, her situation, location and activities, such as navigation or content selection, can be tracked. The collected data from the web usage is saved on web server logs and contains hidden information about user habits and preferences. In some cases, some personal user data

also might be obtained explicitly. Analysis of the data reveals individual characteristics and requirements that is instrumental in user modeling for the personalization of the system recommendations. However, gathering and processing the data using traditional techniques might raise some challenges including scalability, processing time, and low accuracy of learning techniques.

The efficiency of designing user interface is a neglected topic among recommender system practitioners. There are many aspects of system interface that may affect users' opinions, such as the rating visualization and scale, recommendation sequence visualization, justification of recommendation, user trust, the number of recommendations to display, the most appropriate location on the search results' page for the recommendations, and display of predictions at the time users rate the items. Furthermore, interfaces to better explain multi-criteria recommendations should be explored. In particular, user understanding of proposed recommendations that tackles the *transparency* issue (see Table 2) is an important topic to be explored in the context of multi-criteria recommenders [123].

Gamification, using the game design and elements in a non-game context, can increase the user's engagement and retention in interactive recommender systems. To promote user experience, the whole system can be designed like a game user-interface, or using game-like features such as points and penalties. In a recommender system, the gamification aspects can include, but not limited to, providing points for the new ratings/reviews, competing with the other users in order to get more points, designing the system with interactive objects such as 3D representations and animated characters [124].

Security layer

The main advantage of centralized architectures for recommender systems over decentralized ones is due to the security issues of distributed approaches. The growth of user demands from small-scale devices initiates an urgent need of investigating security vulnerabilities in decentralized and peer-to-peer environments. In some approaches, CI has been applied to preserve the users' privacy in distributed architectures by storing the rating at the user side. In the research by J. Zou et al. [14], a semi-distributed Belief Propagation is used by first formulating the item similarity computation as a probabilistic inference problem on the factor graph. This probabilistic inference problem was solved with Belief Propagation as a probabilistic message passing algorithm.

Cryptographic methods, obfuscation, perturbation, and probabilistic methods have shown a powerful privacy preserving effect. Even though they suffer from some drawbacks, such as high computational cost and low prediction accuracy. Furthermore, defending against attacks might cause some privacy loss. When a recommender system tries to keep the least information about its users, there can be items with sufficiently few ratings to be vulnerable to highly-effective attacks. More research needs to be conducted on the tradeoff between security and privacy of the users and proper frameworks to address the issue. To prevent profile injection known as *shilling attack* (see Table 2), common methods for detecting automated agents, such as noting patterns in profiles or user names and the source or speed of account creation, might be employed. It is also necessary to determine how much data collection is enough for the users to preserve a balance between the sacrificed privacy and the gained quality of recommendations.

Evaluation layer

The evaluation layer involves all the layers directly, except the market strategy layer which is affected by validation and evaluation results indirectly. A recommender system performance depends on different aspects, such as the employed algorithms, visual and interaction design, and style and sequence of presentation. The evaluation metrics defined in our taxonomy are applied to each layer. For example, accuracy, serendipity and coverage should be tested in the recommendation core layer, while adaptability and trust are addressed in the interaction layer, and privacy in the security layer. Data warehouse and Business Intelligence (BI) tools such as dashboards, reports and OLAP can provide useful information about the users behavior to monitor the performance of recommender systems [125]. The system administrator can analyze charts containing data from the acceptance or rejection of the recommendations by users.

Human aspects of recommender systems are unpredictable and very hard to assess. Evaluating a recommender from system aspects such as algorithm accuracy and coverage is not difficult or problematic, but evaluating the user trust and perception about the system and recommendation quality or usability is not an easy task to perform. In fact, it can open many research directions towards user perceptions in the implicit evaluation process. Evaluation can include the user privacy concerns, knowledge domain and current emotional state in addition to their preferences.

What makes the recommender system evaluation even harder is the correlation among criteria. The problem is not only about maximizing the performance for each criterion individually, but it is necessary to investigate the system behavior considering all the criteria at the same time and have a balance performance. Azam & Yao [126] tried to consider a balance threshold between two properties, namely “accuracy or appropriateness of recommendations” and “generality or coverage of

recommendations". They applied the game theory model to 1) determine a trade-off between multiple cooperative of competitive criteria in a probabilistic rough set model; and 2) propose a balanced solution between accuracy and generality. However, investigation of criteria interdependency remained unexplored. Finally, the future research directions discussed in this section are summarized in Table 6.

6. Conclusion

In this paper, we reviewed and investigated the development of recommender systems from research and engineering perspectives. The main motivation of this study is to draw the researchers and practitioners' attention to the alternatives during the engineering and development process, identify the weaknesses and strengths of each technique, and introduce new research questions and challenges for the new generation of recommender systems.

System features	*Future research directions*	*Recommended techniques*
User-provider participation	– Investigating the economic benefits and business and pricing models of a recommender system – Analyzing which part to be subsidized and which part to be charged (item provider and consumer) – Rewarding users to rate the items – Studying the recommendation quality from users' perspective	Game theory Two-sided market Network effects
User data/ preference modeling	– Managing uncertainties of preference modeling – Extracting implicit information about the users from their daily activities – Storing data in ontology-based repositories and discovering semantic similarities and relations – Exploring alternate options of ranking and recommending items to the users by considering several criteria	FS, ANN, GA, SI Deep learning Semantic web ontologies Web usage mining MDCM Multi-objective optimization
System platform	– Utilizing distributed and elastic platforms	Cloud computing
System architecture	– Studying mobile applications and security vulnerabilities in decentralized environments	Intelligent agents AIS

Adaptivity	– Designing a system to operate within dynamic environments and autonomously choose the appropriate recommendation algorithm	Autonomous and self-directed learning
User interface	– Experimenting visualization features including rating visualization and scale, recommendation sequence, number and justification of recommendations, location of the recommendation on the page – Exploring the gamification possibilities to increase the user's engagement and improve the usability	Multi-criteria interface design Gamification
Security and privacy	– Analyzing the required amount of user data – Exploring the tradeoff relation between security and privacy to preserve a suitable balance	Data mining Machine learning
System performance	– Exploring user perception by considering user's privacy concerns, experience, knowledge domain and emotional states – Investigating the usability of the system – Studying the criteria interdependencies	Data warehouse Implicit evaluation techniques Game theory

Table 6. Future research directions in recommender systems

The survey revealed that to make recommendations, the easiest way to obtain the data is to simply ask the users. However, the implicit data can make a considerable difference in recommendations and offer a great opportunity to widen this research area. In fact, advanced techniques and practices to discover hidden knowledge about users are yet to be advocated. The survey also revealed that there is no standard or single perfect technique or algorithm to be used in recommender systems. They all have strengths and weaknesses with a potential tradeoff between different criteria. Depending on the application domain, data and facilities, the suitable techniques and algorithms must be chosen. A common threat is that in most cases, there is a need to combine different similarity measures and recommendation techniques to gain peak performance. Nonetheless, collaborative filtering had received the most attraction from the researchers so far mainly due the availability of real-world benchmark cases and the simple structure of the data to be analyzed for producing recommendations.

There is a comprehensive list of introduced metrics, but there is no mature study investigating them. This issue leaves the door open for further research to improve the performance of recommender systems and make it possible to take them from theory to

practice. The widely used and evaluated metric is accuracy even though it is criticized regarding its inability to determine whether the system can recommend valuable items to users, particularly those which are unknown to the requestor.

A broad range of applications successfully benefited from the advantages of recommender systems. Hybrid, collaborative filtering, and content-based systems are the most utilized as they are the most advanced ones from the research foundations perspective. However, there is still a considerable gap between real-world practices and research applications that need further investigations.

This survey had a thorough insight into the next generation and future directions of developing recommender systems. In recent trends, specific attention was given to computational intelligence to make recommendations more interesting and useful. We discussed these advanced techniques in a layered framework containing market strategy, data, recommendation core, interaction, security and evaluation. The inspiring directions, which come with computational intelligence, make this emerging research area more interesting and appealing for further research, development and practice.

Acknowledgements

This work was supported by Natural Sciences and Engineering Research Council of Canada through Vanier Canada Graduate Scholarships and Discovery Research Grant and by Bi-nationally Supervised Doctoral Degrees through Deutscher Akademischer Austauschdienst (DAAD), Germany.