Centre d'Etudes Doctorales : Sciences et Techniques de l'Ingénieur

N° d'ordre 39 / 2017

THESE DE DOCTORAT

Présentée par

Mme: Hdioud Ferdaous

Spécialité : Informatique

Sujet de la thèse : Industrial Recommendation Systems in Big Data Context: Efficient Solutions for Cold-Start issues

Thèse présentée et soutenue le 11/11/2017 devant le jury composé de :

Nom Prénom	Titre	Etablissement	
Ouçamah Mohammed Cherkaoui Malki	PES	Faculté de sciences Dhar El Mahraz- Fès (FSDM)	Président
Ouatik El Alaoui Said	PES	Faculté de sciences Dhar El Mahraz- Fès (FSDM)	Rapporteur
Oumsis Mohammed	PES	Ecole Supérieure de Technologie–Salé (ESTS)	Rapporteur
Elfaddouli Noureddine	PH	Ecole Mohammedia d'Ingénieurs (EMI) -RABAT	Rapporteur
Benabou Rachid	PH	Faculté des Sciences et Techniques –Fès (FSTF)	Examinateur
Frikh Bouchra	PH	Ecole Supérieure de Technologie –Fès (ESTF)	Directeur de thèse
Ouhbi Brahim	PES	Ecole Nationale Supérieure des Arts et Métiers (ENSAM) –Meknès	Co-Directeur de thèse

Elmoukhtar Zemmouri	PA	Ecole Nationale Supérieure des Arts et Métiers (ENSAM) –Meknès	Invité

Laboratoire d'accueil : LTTI-EST-FES

Etablissement : Faculté des Sciences et Techniques de Fès

Dedications

I dedicate this thesis to:

My tendre and loving parents Naîma and Mohammed

without you, nothing could have been possible,

May God keep you for me and lends you a long life

full of health and prosperity ...

My husand Imadeddine

with whom I shared my hardest moments,

who was always to my sides to listen to me and bring me his support.

God bless you and keep us together until life…

My lovely son Layth

you are my joy and my reason for living…

Acknowledgements

First of all, I am grateful to the God for the good health and well-being that were necessary to complete the research and writing of this thesis.

I warmly thank my supervisors, Mrs. Bouchra Frikh and Mr. brahim Ouhbi, for the confidence they showed me during the last five years. I also would like to express my heartfelt gratitude to them, for their regular academic support, patience, motivation, and enthusiasm. Their advices and guidance helped me in all the time of research and writing of this thesis and have been a key driver for my work. I can't forget their hard times reviewing my thesis progress, giving me their valuable suggestions and made corrections.

I would also like to thank all members of the Jury for having kindly agreed to evaluate my work. Special thanks go to the president of jury, Professor Oussama Cherkaoui from the Faculty of Science Dhar Mahraz (FSDM) and to the examiner of my thesis, Professor Rachid Benabou from the Faculty of Sciences and Techniques of Fez (FSTF). I am honored that Professor Saîd Alaoui ouatik from FSDM, Professor Mohammed Oumasis from the School of science and techniques Salé (ESTS) and Professor Noureddine Elfaddouli from mohammadia schools of engineers (EMI) have agreed to report on my thesis.

I wish to express my sincere thanks to Mr. Elmoukhtar Zemmouri for his availability and support during my moves to the research laboratory at ENSAM, for the installation of workstations and the implementation of my code. As well as, for accepting to be a member of the jury as a guest of honor.

I would to thank profoundly all members of the research team *"WRUM"* for their support, help, advices, many constructive discussions and ideas brought during team meetings throughout the PhD years.

I deeply thank my dear Imadeddine for his patience, his presence and his endless support and sacrifices which are an immense source of energy and enthusiasm.

Lastly, and most importantly, I would like to thank from the bottom of my heart and with great love, my parents who never stop believing in me during all my years of study. Thank you for the sacrifices made to my education, for providing a loving environment for me and for their support and especially their patience.

Abstract

Nowadays, users are inundated with the large volume of data flowing in the web. This information overload makes users unable to locate and to find the wanted information at the right moment, especially when they lack sufficient experience to deal with these immense amounts of data. Lately, sophisticated tools were developed to deal with these emergent challenges, among them we find Recommender Systems (RSs).

This thesis deals with automatic RSs that aim to provide items that fit with users' preferences. These tools are increasingly used by many content platforms to assist users to access to the needed information. In fact, to perform correctly a RS needs to model user's profile by acquiring information about user's interests as the visited content and/or user's actions (clicks, comments, etc). However, modeling these interests is considered as a hard task, especially when the RS has no prior knowledge about a user or an item (cold-start issue). Therefore, modeling user's profile is complex, since the generated recommendations are often far away from the real user's interests. In addition, existing approaches are unable to ensure good performance on platforms with high traffic and which host a huge volume of data.

In order to solve these issues and to obtain more relevant recommendations, in this thesis we made three main contributions: 1) proposing a CCSDW method to compute criteria and items weights to be used during the profiling of new users to better understand their preferences and to tackle the new-user issue 2) presenting a hybrid RS based on a linear combination of CF and an enhanced CB approach using HFSM to compensate the lack of data about new items and to deal also with long tail issue 3) implementing a distributed recommendation engine with Apache Spark to enhance the scalability and response time.

To demonstrate the interest of the proposed approaches, they were evaluated using different data sets in term of coverage and recommendation accuracy. Furthermore, the distributed algorithms were evaluated to validate their scalability in an industrial context.

Keywords: Recommender Systems (RSs), Collaborative Filtering (CF), Content-based (CB), Hybrid, Active Learning (AL), Coefficient Correlation Standard Deviation integrated Weights (CCSDW), Hybrid Features Selection Method (HFSM), Cold-start, long tail, Big Data, large scale.

Résumé

De nos jours, les utilisateurs sont inondés avec le grand volume de données circulant sur le Web. Cette surcharge d'information rend les utilisateurs incapables de localiser et de trouver l'information recherchée au bon moment, surtout lorsqu'ils manquent d'expérience suffisante pour faire face à ces quantités immenses de données. Dernièrement, des outils sophistiqués ont été développés pour faire face à ces nouveaux défis, parmi lesquels on trouve les Systèmes de Recommandations (SRs).

Cette thèse concerne les SRs automatiques, qui visent à suggérer des produits adaptés aux préférences des utilisateurs. Ces outils sont de plus en plus utilisés par de nombreux utilisateurs pour accéder à la bonne information. En fait, pour fonctionner correctement, les SRs doivent modeliser le profil utilisateur en collectant des informations sur l'utilisateur et ses interêts comme le contenu déjà visité et / ou les actions de l'utilisateur (clics, commentaires, etc.). Cependant, la modélisation de ces tâches est difficile. Par conséquent, la modélisation est complexe et les recommandations sont souvent éloignées des intérêts réels des utilisateurs. De plus, les approches existantes sont incapables d'assurer une haute performance sur les platesformes à fort trafic et qui hébergent un volume de données énorme.

Afin de résoudre ces problèmes et d'obtenir des recommandations plus pertinentes, cette thèse comporte trois contributions principales: 1) proposer une méthode CCSDw pour calculer les poids des critères ainsi que des items afin de les utiliser lors du profilage des nouveaux utilisateurs, ce qui permettra de mieux comprendre les préférences des utilisateurs et résoudre le problème du nouveau utilisateur 2) présenter un SR hybrid basé sur une combinaison linéaire de l'approche du Filtrage Collaborative (FC) et une approche basé sur le contenu qui utilise une méthode de selection hybride des caractéristiques (HFSM) pour compenser le manque de données sur les nouveaux articles et pour traiter également le problème des produits à faible demande 3) mettre en œuvre un moteur de recommandation distribué avec Apache Spark pour améliorer l'évolutivité et le temps de réponse.

Pour démontrer l'intérêt des approches proposées, elles ont été évaluées en termes de couverture et d'exactitude de la recommandation, en utilisant différents jeu de données. De plus, les algorithmes distribués ont été évalués pour valider son évolutivité dans un contexte industriel.

Mots-clés : Système de Recommandation (SRs), Filtrage Collaboratif (FC), recommendation basé sur contenu (BC), Hybrid, Active Learning (AL), Coefficient Correlation Standar Deviation integrated Weights (CCSDw), Hybrid Features Selection Method (HFSM), démmarage à froid, produits à faible demande (longue queue), Big Data, recommandation à grande échelle.

ملخص

في الوقت الحالي، يتم غمر المستخدمين بحجم كبير من البيانات المتدفقة عبر شبكة الإنترنت. هذا الحمل الزائد للمعلومات يجعلهم غير قادرين على تحديد موقع المعلومات المطلوبة والعثور عليها في اللحظة المناسبة، وخصوصا عندما يفتقر هؤلاء المستخدمون إلى الخبرة الكافية للتعامل مع هذه الكميات الهائلة. في الآونة الأخيرة، تم تطوير أدوات متطورة للتعامل مع هذه التحديات الناشئة، من بينها نجد نظم التوصية.

هذه الأطروحة تتعامل مع نظام التوصية الأوتوماتيكية، التي تهدف إلى توفير العناصر التي تتناسب مع تفضيلات المستخدمين. يتم استخدام هذه الأدوات بشكل متزايد من قبل العديد من منصات المحتوى لمساعدة المستخدمين على الوصول إلى المعلومات الصحيحة. في الواقع، لأداء صحيح يحتاج نظام التوصية إلى نموذج المستخدم الشخصي من خلال الحصول على معلومات حول مصالح المستخدم مثل المحتوى الذي تم زيارته و / أو الإجراءات التي قام بها المستخدم (النقرات والتعليقات، الخ...). ومع ذلك، نمذجة هذه المصالح تعد من المهام الصعبة في حالة عدم توفر نظام التوصية لمعرفة مسبقة عن المستخدم أو العنصرالمراد توصيته (مسألة الإقلاع التمهيدي). لذلك، النمذجة الشخصية للمستخدم معقدة، لأن التوصيات التي تم إنشاؤها في كثير من الأحيان بعيدا عن مصالح المستخدم الحقيقي. وبالإضافة إلى ذلك، فإن النهج القائمة غير قادرة عموما على ضمان أداء جيد على المنصات ذات الحركة العالية و التي تستضيف قدرا كبيرا من البيانات.

لذلك، ومن أجل حل هذه المشاكل والحصول على المزيد من التوصيات ذات الصلة، في هذه الأطروحة قدمنا ثلاث مساهمات رئيسية:

1) اقتراح طريقة CCSDW لحساب المعايير وأوزان المنتجات التي سيتم استخدامها خلال تكوين سيرة المستخدمين الجدد لفهم تفضيلاتهم بشكل أفضل ومعالجة قضية المستخدم الجديد.
2) تقديم نظام توصية هجين يعتمد على مزيج من مقاربة تشاركية (CF) والنهج المحسن و المعتمد على محتوى المنتجات (CB) للتعويض عن عدم وجود بيانات حول المنتجات الجديدة والتعامل أيضا مع قضية المنتجات ذات الطلب المنخفض.
3) إنجاز محرك توصية على نظام موزع لتعزيز قابليته التدرجية و سرعة استجابته لطلبات المستخدمين.

لإثبات أهمية النهج المقترحة، تم تقييمها باستخدام مجموعة بيانات مختلفة في نطاق التغطية ودقة التوصية. وعلاوة على ذلك، تم تقييم الخوارزميات الموزعة للتحقق من قابلية التوسع في السياق الصناعي.

Contents

LIST OF TABLES:

LIST OF FIGURES:

LIST OF ABREVIATIONS

- **RS:** Recommender System.
- **IF:** Information Filtering.
- **IR:** Information Retrieval.
- **CB:** Content-Based.
- **CF:** Collaborative Filtering.
- **KB:** Knowledge-based.
- **DM:** Demographic approach.
- **AL:** Active Learning.
- **ML:** Machine Learning.
- **AI:** Artificial Intelligence.
- **TF-IDF:** Term Frequency Inverse Document Frequency.
- **VSM:** Vector Space Model.
- **SVD:** Singular Vector Decomposition.
- **KNN:** K-Nearest-Neighbor.
- **MF:** Matrix Factorization.
- **LSA:** Latent Semantic Analysis.
- **LSI:** Latent Semantic Indexing.
- **RDDs:** Resilient Distributed Datasets.
- **CCSDW:** Coefficient Correlation Standard Deviation integrated Weighting.
- **HFSM:** Hybrid Features Selection Method.
- **MADM:** Multi Attribute Decision Making.
- **NDCG:** Normalized Discounted Cumulative Gain.
- **NDPM:** Normalized Distance-based Performance Measure.
- **MAE:** Mean Absolute Error.
- **HDFS:** Hadoop Distributed File System.

GENERAL INTRODUCTION

"Too much information kills information and leads to misinformation"

Anonymous

"Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you"

Greg Linden, creator of Amazon

Research Context & motivations

The Internet is a digital network providing users with a wide variety of resources, also known as "items". The term "item" stands for any type of electronic document containing an informative data accessible in a given electronic format (e.g. Textual or multimedia format), which has the distinction of being heterogeneous and distributed and whose numbers is in a constant growth. In fact, the migration from the traditional world to the Internet has made access to information easier and faster than ever before. However, this beautiful world has created other needs and concerns for researchers who are forced to adopt a new rhythm to survive in this new era.

In this context, the use of tools to facilitate access to relevant items is crucial. Among the first tools that have been developed to address this problem of accessing relevant items on the Web, we find search engines. The content offered by these engines is not personalized, i.e. if the same query is formulated by two different users, the proposed items will often be the same regardless of who conducted the search. This poses a problem, because even if two users express the same query, they do not necessarily have the same needs.

Actually, with the expansion of the Web, taking into account the user during the Information Retrieval (IR) process became a necessity. This allows to meet user's specific needs and thus retaining him for long, especially when the user is well-known beforehand and not occasional. These challenges related to the satisfaction of users' expectations and their retention are the main objectives of the personalized access to information. Indeed, the personalization (customization) is an area of research that captures the attention of many researchers, whose aim is proposing items related to the real tastes of each user.

Recommendation Systems (RSs) are part of the customized access to information. They propose to an active user (i.e. The current user) items that they consider relevant to his expectations. They seek to anticipate their future needs through the prediction of their assessments concerning one or more items that they have not yet rated/consumed. In other words, RSs are designed to assist the user's research activity and to orient him towards the information that suits him best.

In a recommendation process, the identification of users' evaluations (expressed as ratings, reviews, comments, etc.) is often fundamental as it makes possible to know more about the concerned user in order

to propose him relevant recommendations. These evaluations reflect the positive or negative opinions of the user about a certain number of items. Their identification may vary depending on the type of approach used. This primary step (called "elicitation") is certainly a tedious process for the user, since he is solicited in order to express explicitly his interest towards a set of items. However, if this step isn't carried out accurately, any recommendation can be delivered which implies discouragement and the abandonment of the user.

In our thesis, we were interested in the study of RSs in industrial context. We focused mainly on tackling issues related to data sparsity and the lack of sufficient information about either users or items. The following section presents the research questions that we addressed through this thesis.

Research issues

As we have previously stated, RSs are designed to customize access to the information, to this end they use filtering techniques to model users and recommend relevant items based on the opinions of their neighbors. The adoption of filtering systems is quite important, but the challenge is to improve the practices and methods used to make the system more precise, interactive, efficient and adapted to particular contexts. This improvement involves enhancing the management of problems related to these types of systems such as cold start and by proposing new sophisticated approaches to make their functioning better. Different research questions (problems) can emerge from this definition, which can be articulated on the following axes:

- First, the **cold start** issue. Considered as the main issue of RSs, since it is impossible to generate recommendations when no data is present. Indeed, how could we recommend something to a user on which we have no information? How to recommend a resource (an item) whose nature is unknown? It is necessary to have alternative sources, semantics, available at the start to be able to attenuate the effect of this problem and to bootstrap a RS newly launched.
- Second, **the quality of prediction**. Even after startup of a RS, the quality stills low, as the evaluation matrix is often sparse (<5% evaluations for Netflix and Movielens). Hidden information must be sought with advanced statistical methods, or semantic information can be used to mitigate the low number of evaluations.
- Third, **the performance and scalability of the system**. With the current concurrency, users are pressed to get what they want very quickly. Hence, it is not acceptable to afford recommendations for a user on the web in more than one second. So certain methods, even they are very precise, cannot be applied if they do not respect the criteria of performance and scalability.

The interaction of the user with the system is also an important criterion, even if the user is not satisfied with the recommendations, the system must at least offer him the possibility to easily navigate and find the items that he looks for. Encouraging him to evaluate resources is also important.

The contribution of the thesis

The contributions of this thesis include:

State-of-the-art of RSs: one among the objectives of this thesis is conducting a detailed study of state-of-the-art on RSs, in order to distinguish the strengths and weaknesses of recommendation techniques in its different forms (i.e. Collaborative Filtering, Content-based and hybrid) and finally to identify best practices, known and proven that achieve the best recommendations in term of high quality.

Resolution of new-user cold-start issue: in order to tackle this issue, we proposed an Active Learning (AL) process based on the method of weighting items that we called CCSDw. This method is rested on the exploitation of multi-criteria ratings to evaluate the importance of criteria and compute their weights. Then, criteria's weights are used to compute the items' weights and reorder them to be subsequently rated by the new user during the profiling process.

Resolution of new-item cold-start issue: another important issue, on which we focused, is the new-item issue known also under the name of first rater problem. We handled this issue by using a new content clustering algorithm based on Hybrid Features Selection Method (HFSM). This method allows to compute similarities among items based on their content instead of their ratings, this allows to compensate the missing ratings either in case of items belonging to the long tail or for the newest ones.

Distributed recommendation platform in a Big Data context: Our last contribution aims to integrate all the techniques and discussion of previous contributions in a distributed recommendation infrastructure. To this end, we exploited techniques used in Big Data context- more precisely Apache Spark- to implement the proposed algorithms. The choice of establishing a Recommendation Engine on a large scale by Spark can offer a high scalability and minimal response time which leads to satisfied users. The empirical results are presented to compare the effectiveness of our approach to different recommendation algorithms.

Organization of the manuscript

In addition to general introduction and conclusion, this thesis dissertation is organized as follows:

The first part that brings together Chapter 1 and 2, presents a detailed study of state-of-the-art of Recommendation Systems (RSs) in industrial Context:

- **Chapter 1** defines the context and motivations of our work, namely the access to information. The two main research axes considered by the literature to assist the user in accessing relevant information, namely Information Retrieval (IR) systems and Recommendation Systems (RSs), are described. Then, we emphasis on RSs and particularities of these systems are highlighted.
- **Chapter 2** presents a detailed study of state-of-the-art about RSs. The chapter presents the different types of such systems. The issues and challenges related to each type of recommendation are then discussed. Furthermore, the existing evaluation methods are specified to which we refer throughout the manuscript. Finally, a brief overview of some of the most common real-life RSs takes place at the end of the chapter.

The second part which covers Chapter 3, 4 and 5 is dedicated to present our contributions:

- **Chapter 3** describes our proposed method to tackle the new user cold-start issue, which consists on conducting AL process based on a set of items selected by the CCSDW method (a method for weighting criteria and items rested on multi-criteria ratings). Our proposition is evaluated through a user experience, where we also highlight the interest of minimizing user's effort and maximizing the quality of the computed predictions for the satisfaction of the user.
- **Chapter 4** tackles another main issue related to RSs, namely new-item cold-start problem. We proposed a hybrid recommendation method rested on content clustering with Hybrid Features Selection Method (HFSM).
- Finally, **Chapter 5** presents the implementation of the proposed solutions using Apache Spark architecture, which is suitable in a real industrial context which is subject to heavy traffic. After defining the constraints posed by this context, as well as the tools put in place to evaluate the performance of our model, we discuss the results obtained which demonstrate the viability of our approach.

Part I
State-of-the-art

Recommendation System in industrial context

1
BACKGROUNDS AND SCOPES

In this age of information overload, people use a variety of strategies to make choices about what to buy, how to spend their leisure time, and even whom to date […]

Dietmar Jannach, 2011

SUMMARY

1.1. Introduction

This first introductory chapter presents the motivation and purposes behind the emergence of a new generation of systems and how accessing to information have evolved in few last decades. Then, it discusses the key points of automatic Recommender Systems (RSs) and their advantages compared to classical existing systems.

In a first step, we identify the core functions of the RS. We present also the different data sources that can be in use during recommendation process. Then, we will list the different classifications of automatic RSs. We distinguish between the usual types and extended ones, particularly through a functional typology. Finally, we give different steps of the recommendation process and an abstracted model of the whole process.

This chapter will allow us to focus on the state of the art of some specific basic techniques in relation to the industrial context.

1.2. Context and Motivations

1.2.1. Information access modalities

The high availability and the variety of information on the Internet are inundating users [1]. This information overload - instead of being beneficial- drives users to make poor decisions, especially when they lack experience and skills to evaluate different information. *"Getting Information off the internet is like taking a drink from a fire hydrant"* - Mitchell Kapor. In fact, content would be wasted if that information could not be found, analyzed, and exploited correctly. On the other side, each user should be able to quickly find information that is both relevant and comprehensive for his needs.

The general purpose of information access systems is establishing a win-to-win relationship between users and service providers by allowing these last to implement effective tools to access information in order to satisfy the user and retain his loyalty. Hence, modes of accessing this information evolve to match these new requirements. In the literature, we distinguish between two ways of information access:

Active access [2]**:** In which the user has to make an effort to find what s/he wants, e.g. Navigating hypertext documents (by following links), navigating through the categories of a directory information (Yahoo, Google Directory) to target the searched resources, or by formulating his needs explicitly as a query by using search engines (Google, ...). This way of acceding information is provided by *research systems*, called also *Information Retrieval (IR) systems*.

Passive access [3]**:** In this case, the user receives *automatically* information or suggestions on what would interest him without any intervention from his part. As a result, the required effort to obtain the new information is reduced. For instance the recommendations made by a friend, a mailing list or by filtering system information (Amazon, Google news). This way of accessing information is ensured by an *Information Filtering (IF) system*.

1.2.2. Information Retrieval (IR)

Information Retrieval (IR) systems allow the user to search within documents collection, by articulating his needs as a query, usually expressed in natural language. The documents returned are ordered according to their degree of relevance to the expressed query. Even if the use of such type of tools seems to be familiar to users, but the formulation of their needs as a query is often complex and can lead to vague and ambiguous or too specialized requests. The information retrieval process can be modeled by the so-called U-model given in Figure 1.

The documents collection is analyzed during the indexing step in order to produce a representation that can be interpreted later. On the other hand, user's needs are formulated by the user himself as "query" that is expressed in a natural language and which is analyzed in its turn in an analogous process to indexing step, in order to have a similar representation of documents. Then, the query and indexes are matched by the mean of measures matching. These measures are almost of time similarity metrics, used to find the degree of resemblance between the representation of documents and that of the query; as a result a set of matched documents is returned. The relevance of returned results is evaluated by the user himself, who may reformulate his query if the results don't meet with his needs.

However, classical IR technologies present many limitations. First of all, IR systems require a certain commitment from the user since the user's query triggers the retrieval process. Using IR systems can be effective when the user knows exactly what he is looking for. But in real cases, users may not know what to search or how to reformulate his query; in addition, he is not aware of the range of all available options and when he is overwhelmed with a large number of results. So, he became unable to make a good decision about what to pick.

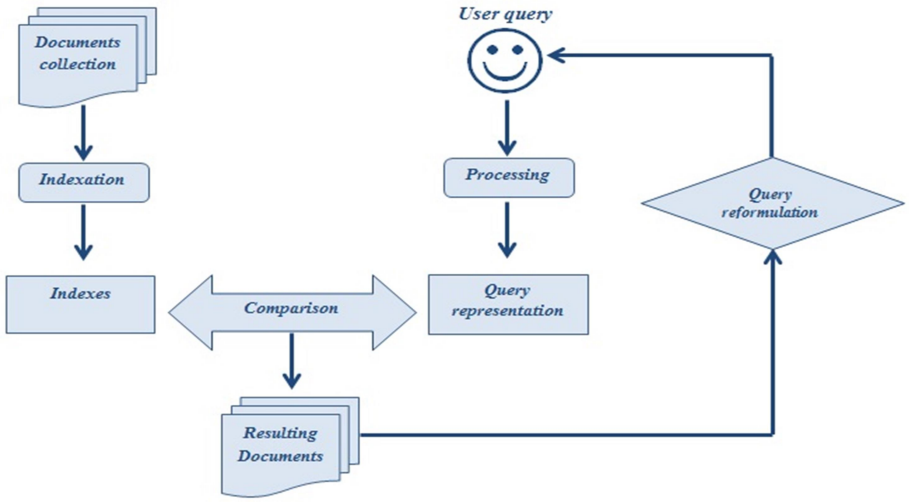


Figure 1: Information Retrieval process.

Users are in need of more sophisticated tools than a simple IR system, that are able to make decisions for them and keep them informed about topics on which they are interested with the minimum effort. Information Filtering (IF) systems bring the solution to these issues by suggesting *automatically* documents to users.

1.2.3. Information Filtering (IF)

An Information Filtering (IF) system assists users to find what they want exactly, by filtering out irrelevant information and keeping only those that seem interesting and useful for them. This type of systems is suitable to manage the information overload, by removing all redundant or unwanted information from an information stream, unlike IR systems that deliver all information related to the user's query and let him decide what to select from the information stream or refine his query. In order to identify user's interests to be able to filter information based on them, IR system requires building a user's profile by using feedback from the user about his preferences.

Generally IF systems are considered as usual IR ones, with the difference that the user's query- expressing his preferences- is not formulated by the user himself. Instead, personal needs/ preferences are inferred even explicitly or implicitly, and then are stored in a user profile that expresses his long term trends. Then, this profile is compared against information to filter out those corresponding to user's needs.

In fact, the strength of IF systems resides in their dynamicity; the information stream is generated dynamically and doesn't require any intervention from the user. The delivered information comes in different forms (e.g. Alerts, emails, etc.), especially when they are on the form of suggestions the IF system is called a Recommender System (RS). The figure 2 presents the general information filtering process.

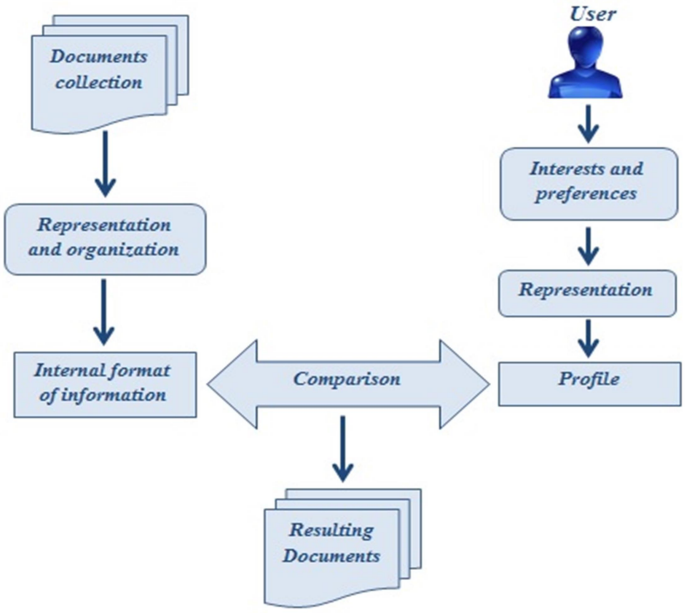


Figure 2: Information Filtering process.

1.2.4. IR vs. IF

Generally, IR and IF systems have many commonalities and differences which make them complementary [3]:

- Seeking for particular information using an IR system refers to a *singular* use of the system by a user that has a purpose and a query at a time, whereas, IF process refers to repetitive uses of the system by an individual/ group of users having long-term interests each time.
- IR systems are responsible of collecting and organizing information following their *request* according to its degree of relevance, while IF system distributes information to groups or individuals *automatically*.
- The IR system selects documents from a relatively static database. In contrast, an IF system selects or removes documents from a dynamic stream of data.
- The retrieval has some issues with matching queries to information needs. The filtering assumes that the continuous updates on users' profiles can compensate these issues.
- IR deals with large repositories of unstructured content about a large variety of topics, while IF focuses on smaller content repositories on a single topic.
- The personalization aspect in IR systems doesn't receive big attention (we hadn't heard up to now about a personalized search engine such as Google), yet they could reorganize information based on recent research done on learning to rank. In addition, IR used some idea coming from IF systems (e.g. A given page is important when it is related or endorsed by others).

- Using an IR system allows only to locate a possible relevant content and the user himself is responsible to evaluate if the results are good / relevant enough. When it comes to IF system, it considers that the user has no knowledge to evaluate content relevance, hence the system works on differentiating between relevant content.
- The IR allows the user to interact with the document during a single search session. On the other hand, IF allows long-term changes through research session series.
- Both of IR and IF support different stages of the information search/discovery process.

Table 1: Brief comparison between Information Retrieval (IR) and Information Filtering (IF) systems.

	Information Retrieval (IR)	***Information Filtering (IF)***
The system requires user's request?	- Yes - It is mandatory - The user's query is expressed explicitly	- No - It is optional - Or it's acquired implicitly and stored in profile
The user knows exactly what he wants?	Yes	No
Repositories' Content	Large and unstructured	Small, on one topic
Main functions	Research and Exploration	Discovery and Navigation
User involvement	High	Low
Input data	User's query	User's profile
Personalization	✗	✓
Dynamicity	✗	✓
The systems pushes content into users?	No	Yes
Main used techniques	Content-based (CB)	Collaborative Filtering (CF) and Content-based (CB)

1.3. Recommender Systems (RSs)

1.3.1. Definitions and Terminologies

In everyday life, people base their choices and decisions on the recommendations of other people that can take different forms. For instance, they may be either by word of mouth, reviews printed in newspapers or recommendation letters. Developers were inspired from this simple idea to develop recommendation algorithms that mimic this behavior automatically, in order to support customers to find what they want with the least effort. The first emergence of Recommender Systems (RSs) was in the middle of 1990's, to cope with issues related to information overload.

RSs [4] are a subclass of IF system; they are defined as software tools with techniques that provide suggestions of items that can be in use by users, helping them to make good decisions. Generally, the recommendations can take two main forms:

- *Non-personalized* (for anonymous users): represents the first recommendation algorithm which is the most simple to generate but may be unuseful in some cases, since the recommendation is the same for all users. The obvious example of this category is the Top-N recommendation that is normally featured in magazines, websites or newspapers.
  The recommendations are either manually selected by the online retailer, based on the popularity of items (average ratings, sales data, total visits) or the recommendations can be the top-N new products of the e-shop.
- *Personalized*: that use RS to personalize the online store to each user. In this case, the RS tries to predict what items are more suitable to the user based on his *preferences* expressed explicitly or inferred by his behavior implicitly i.e. his interaction with the system when he is online. This type of recommenders is widespread and there are many examples of them, like amazone.com, Google news, Netflix, MovieLens, YouTube, etc.

Formalization of recommendation problem:

Generally, a RS seeks to predict either the rating or the preference that a user would give or have for an unseen item. Formally, a RS can be seen as a utility function $\hat{R}(u,i)$ that predicts the likelihood (degree of utility) of an item i for a given user (active user) u_a [5]:

$$R: U \times I \rightarrow \hat{R}(u,i)$$
$$(u_a, i) \rightarrow \hat{r}_{u_a,i}$$

The recommendation problem can be divided into two sub-problems:

1) Prediction problem: whose aim estimating the item's likelihood for a given user, the prediction is computed for unrated items i by the active user u_a, i.e. $i \in I \setminus I\,u_a$ since $I\,u_a \subseteq I$.

2) Recommendation problem: based on prediction function, the RS in this case recommends a ranked set of K items ($K \ll N$), according to the predicted value $I_{recommended} \subset I$ that the user u_a will like the most. The recommended items shouldn't be from user's interest, i.e. $I_{recommended} \cap Iu_a = \emptyset$.

1.3.2. Functionalities and area of use

The use of RSs is not only beneficial for users who utilize these systems to find and discover good items, but also they can play an important role for service providers. A RS allows them better identifying and understanding user's needs, as a consequence, they may suggest them diverse items that fit with their tastes (not only popular ones), which increase the number of items consumed / purchased by users. The more these latter are satisfied, the more they use the RS and they become faithful to him.

In [6], Herlocker et al. 2004 have defined eleven goals and tasks for which an end-user uses a RS that are depicted in Figure 3:

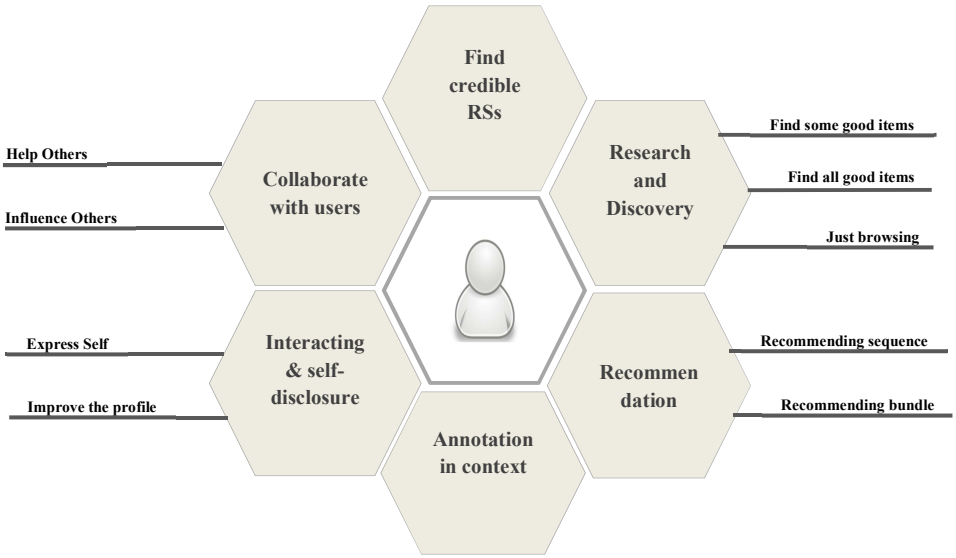


Figure 3: Different tasks and goals of a RS.

The Figure 3 presents various tasks [7], among them we find those that are fundamental core tasks of RSs which affect the end-user directly, such as recommending relevant items, discovering interesting content. Other tasks are considered as "opportunistic" ways to use a RS, like expressing self that helps to refine user's profile and deliver more accurate recommendations, as well as users' interactions from which the RS benefit to recommend items to other people (collaboration aspect). The variety of these tasks points to the fact that RS's role may be diverse, according to the context and the area of the use that are uncountable.

On its beginning, the use of RSs was confined in commercial applications. However, there are many possible application domains in which recommendation algorithm may be adopted such as e-commerce, e-learning, entertainment, services, etc [8, 9, 19]. This diversity of application domains implies a diversity of recommendations scenarios. Subsequently, it calls for exploiting different types of data sources and techniques depending on the domain requirements. The next sub-section is dedicated to discuss different types of input data, required during recommendation process in order to generate good recommendations.

1.3.3. Required data source (Input data)

Automatic RSs are Information processing systems that require essentially different kind of knowledge and data for their functioning. The type of required data may vary from one technique to another, it depends merely on the recommendation paradigm used (see next section). Generally, data used during the recommendation process are related primarily to three kinds of objects (What- Who- How) [10]:

- **Items:** denote *what* type of product will be recommended (RSs are specialized, e.g. books RS, movies RS, etc.), is characterized by their properties (called also metadata or features), complexity and value of the utility (that can be either positive or negative) which represents the level of appreciation by the user.
- **Users:** people *who* use the RS in the hope to find interesting items. In order to benefit from personalized recommendations, users are characterized by a model (user profile) which gather their data and preferences. This later is constructed differently depending on the techniques used. For instance, in the case of CF recommenders user' profile contains ratings, but for demographic-

based RSs the user profile is constructed based on using socio-demographic data (e.g. Age, sex, location, etc).

- **Transactions:** refers to recorded interactions between user and RS which save important data characterizing the relationship between pairs of users and items. These transactions are exploited by the RS to generate recommendations.

The whole of these data is used by the RS according to the followed approach, some approaches need information about users, others use items' features and some other approaches require both of them. However, it is not the matter of the use of these data as far as concerns the way how the RS acquires these knowledge and data sources. Generally, acquiring these data is ensured in two different ways, either explicitly or implicitly.

Explicit data: This category of data is requested explicitly by the RS and provided by the user himself. The advantage of this method of acquiring data is the reliability of the obtained information. However, explicit data are almost of times very scarce, since users are reluctant to express such private information. Explicit data can be in different forms, here are some examples of them:

- Customer Feedback: Numerical Ratings, binary ratings, textual comments, etc.
- Demographic data: profession, sex, age, etc.
- Physiographic.
- Ephemeral Needs.

Implicit data: Can be inferred automatically by monitoring user's behaviors and analyzing it later, among these data we find:

- Purchase History.
- Click or Browse History (navigation).
- Session duration.
- Number of visits.

It is worth to note that in the industry, logs of navigation and logs of purchases are more frequent than logs of ratings.

1.3.4. Different Classifications and typologies

Recommender Systems (RSs) are a broad research field which gathers many techniques and approaches [11]. These systems have taken techniques from the IR field such as content-based filtering, as well as they use the Collective Intelligence principles and other data mining and probabilistic tools. All of these techniques are used on the aim to predict which item is relevant for the user and subsequently recommend it, but the manner with which a RS carries out the process defines the type of recommendation used. There are many classifications and types for RSs, the two main classifications are the following (see Figure 4):

- ***Traditional classification*** (Adomavicius and Tuzhilin, 2005) [5].

Categorizes RSs into three main approaches: Content-based (called also thematic-based), Collaborative Filtering (CF) and Hybrid approaches:

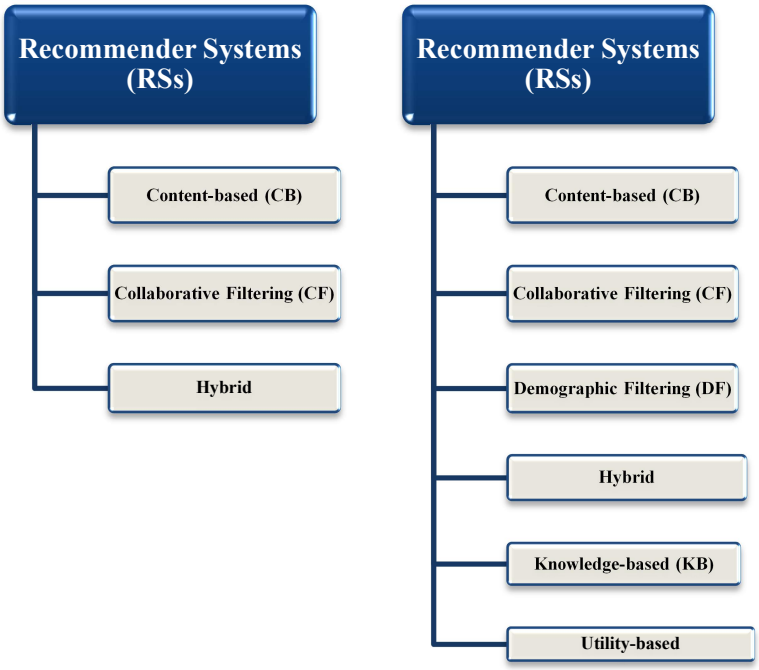


Figure 4: The two main classifications of RSs in the literature.

- ***Extended Taxonomy*** (Burke 2007; Rao and Talwar, 2008).

Assuming that there is a variety of knowledge data about items, users and transaction leveraged in recommendation process, the extended taxonomy proposed in [12, 13] distinguishes between six different types of RSs based on input data required:

- **Content-based:** the system recommends items which are similar to those that were liked by the user in the past. The similarity is calculated from features and properties of the item, e.g. in the context of movie recommender, if the user profile indicates that his own likes or/and prefers comedy movies, the RS will often suggest to him all comedy movies that exist, given that a genre is among the features that characterized movies.
- **Collaborative filtering** [14] also called *Community-based* [15, 16]**:** based on the Principles of "Tell me who your friends are, and I will tell you who you are" and "mouth to ear". It recommends to the active user items that similar users had liked in the past. The similarity between users is based on the ratings history of the users (people-to-people correlation). There is also a special sub-type of *Collaborative approaches*, based on demographic data about users, this type of RS is called **Demographic Recommender** [17], and which operates based also on correlations between users, by using demographic data (age, occupation, location, etc.…) instead of ratings.
- **Knowledge-based (KB)** [18]**:** recommends items based on a specific knowledge domain, the features of items that go with a user's preferences and constraints. Another special case of KB approaches is the **Utility-based** [13] which relies on items' features to calculate utility function for each item for the user. The computation of this utility function is based on functional knowledge.
- **Hybrid RS** [13]**:** combines the whole of techniques already aforementioned in different ways to meet the requirements and needs.

The detailed state-of-the-art of these different approaches will be studied in the next chapter, in which we give the general approach and techniques used for each one.

1.4. General Recommendation process

In the main, to produce items' recommendations each RS follows a specific recommendation process. Like any other process, this one requires input data that have to be communicated by the user, in the aim to express his preferences. Once this information is acquired, the RS conducts a user preference learning phase, which is considered as an interpretation step where a user's information is somehow transformed into a *model* (presents how users' preferences are understood and depends merely on the recommended technique employed). The generated model is used to infer personalized recommendations with respect to the expressed preferences.

The Figure 5 shows a general view of how the recommendation process is performed in a sequential manner. The figure presents the recommendation process as an *interactive* and *iterative* process at the same time. The presented recommendations are in perpetual change as and when the user gives his feedback about the presented items. This continuous evaluation allows updating user's model (i.e. Profile), until that the user is satisfied or get bored.

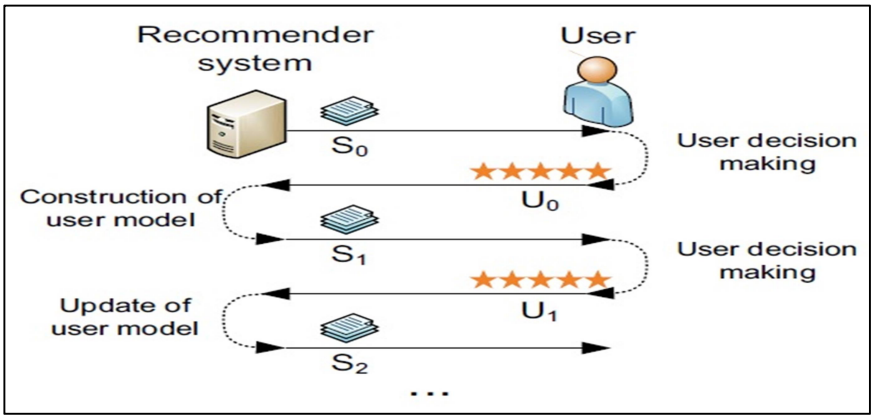


Figure 5: Sequential communication between the user and RS.

To sum-up, here are the six main steps according to which the recommendation process is carried out (Figure 6):

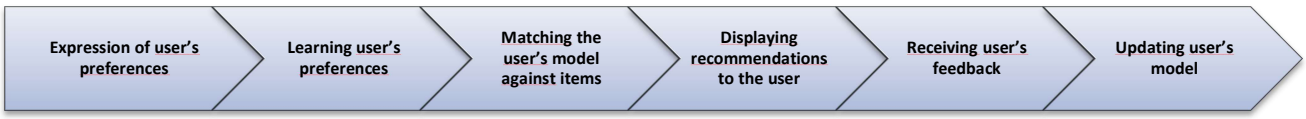


Figure 6: Main steps to carry out a recommendation process.

The user performs some actions with the system, which are processed by various components of this latter. Furthermore, some inputs may be processed by multiple components. The figure 7 depicts the whole recommendation process with the different possible components that vary according to the type of input and the subsequent processing:

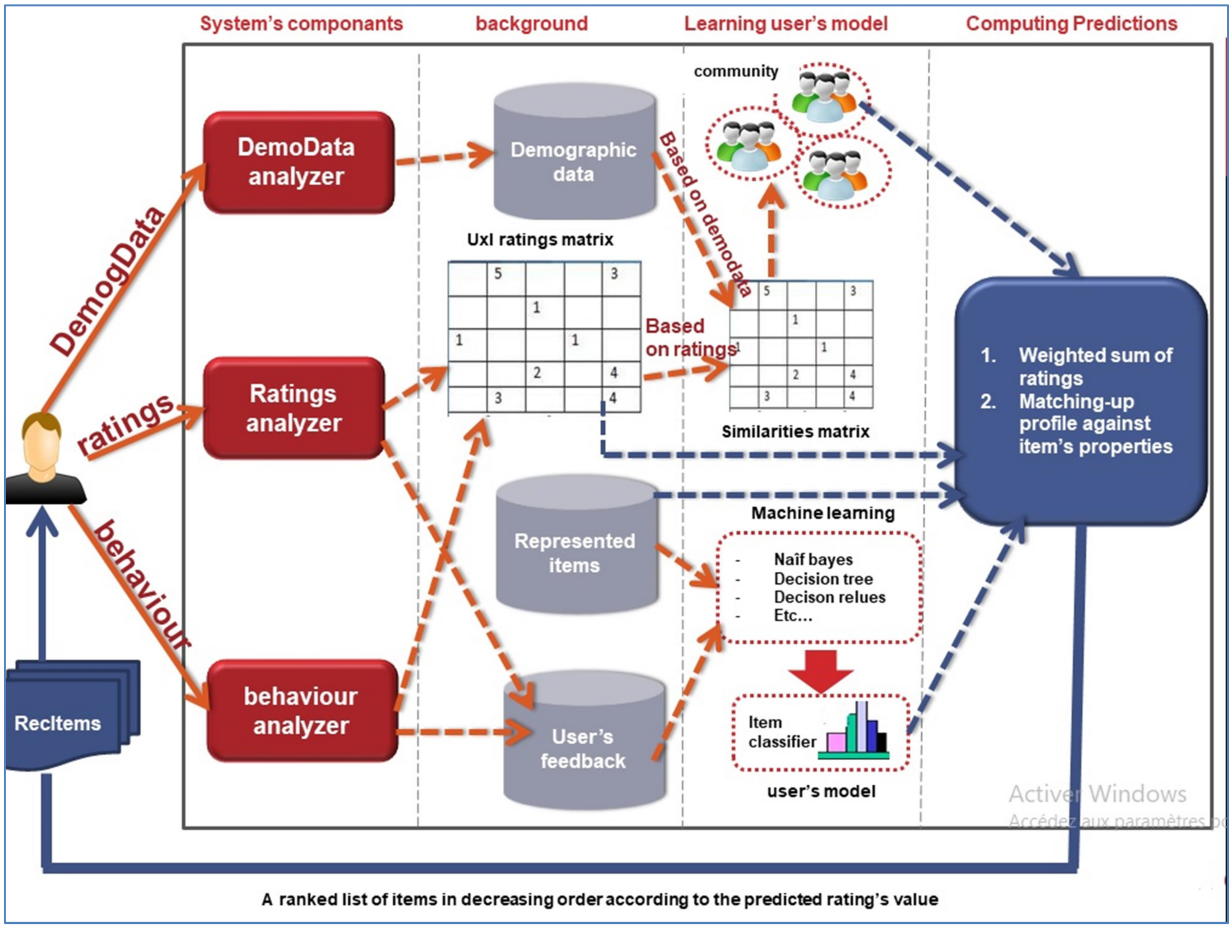


Figure 7: Abstracted Model for recommendation process.

2
STATE-OF-THE-ART

The recommendations provided are aimed at supporting their users in decision-making business processes […]

Dietmar Jannach, 2011

SUMMARY

2.1. Introduction

This chapter deals with the state-of-the-art of Recommendations Systems (RSs), which covers reviewing different techniques used in this regard, different methods of evaluation, challenges and issues comforted in RSs. We end with a brief review of some representative RSs in a real life that we use almost every day without realizing that there is a recommendation algorithm behind its functioning.

We distinguish between three main classical paradigms for recommendations, which are extensively employed in research and real contexts: Content-Based (CB), Collaborative Filtering (CF) and Hybrid. These techniques are discussed in more details. Then, other techniques like Knowledge-based, demographic-based, Context-based, and RSs based on sentiment-analysis are addressed briefly. Thereafter, we cover the most prominent issues and challenges faced in recommendation techniques which must be handled to build an efficient RS in industrial context.

2.2. Review of classical Recommendation approaches

2.2.1. Content-based (CB)

2.2.1.1. General approach

The Content-Based (CB) Filtering is one of the most important types of filtering systems, which is rested on content. This approach has its roots in both of Information retrieval (IR) and Information Filtering (IF) research fields. Since the RS's aim is offering relevant and interesting items to its users, it's more rational to infer recommendations that are *similar* to what they preferred/consumed on the *past* (i.e. Give me more of what I prefer or more of the same). Hence, content-based recommenders require the availability of detailed items' description (innate attributes of items) and user's profile (i.e. Structured set of user's preferences that can be established manually or inferred automatically). Then, items' features are matched up against the concerned user's profile to generate recommendations, using a *similarity function*.

Content-based techniques are a domain-dependent algorithms, since they focus mainly on the analysis of the attributes of items to infer good predictions. The early systems focused on the text domain, and applied techniques from IR to extract *meaningful* information from the text. The use of such techniques is suitable, when documents like web pages, publications and news are to be recommended [19]. Yet, recently have appeared some solutions that cope with more complex domains, such as music. This has been possible, partly, because the multimedia community emphasized on and improved the feature extraction and machine learning algorithms. Such approaches are essential for bootstrapping RSs, but are not effective for cross-selling recommendations.

2.2.1.2. Recommendation process based on CB approach

As the name implies, the inference of personalized recommendations using the CB depends merely on the content data of items and user's profile which gather his past preferences and tastes [20, 21]. The recommendation's process implementing the content-based approach, is carried out in the following three steps (Hdioud et al. 2013):

1) *Item representation*: in which a set of items is analyzed, by using information source coming from items' description, that are treated to extract features and finally produce structured items' content. At the end of this stage, each item is represented as a point in *n* dimensional space with *n* features which can be numerical, categorical or textual. Then, these features are stored to be exploited later during the process.
2) *Learning a user profile*: it consists in building a model corresponding to the active user, based on the items that he had purchased, liked or rated in the past. To this end, two types of inputs are required:

- **Feedback:** user's reactions against items are called feedback (annotations), and it serves to construct and update user's preferences in the profile of an active user. The feedback can be either implicit which is inferred automatically by monitoring user's behaviors or explicit where the user is asked to express his appreciation towards an item, in various ways:
  - Like / dislike: binary feedback, the user evaluates the items as being relevant or not for him.
  - Ratings: the user assigns a score to an item, based on the rating scale offered by the system.
  - Comments: the user expresses his appreciation with a text without imposing on him, the way to do it. This method seems to be the most complicated, as it requires an intelligent system, which can interpret the text and decide if the comment seems to be a positive or a negative feedback.
- **Training set TR_K**: it's a set of pairs <I_k, R_k> that are extracted from the "represented items repository", where R_k is rating of the item I_k provided by the active user u_a.

Then, feedback and the pairs are gathered to be processed and to generate a user profile.

3) *Recommendations' generation*: this final step consists in matching up attributes of the user's profile against items' features, to suggest the most likely interesting items L_a (List of recommended items to the active user). Thereafter, the system generates a result as a level of user's interest for the new items, in order to filter them if they are appreciated by the user in question, or preventing them from being appeared if not.

Each of these steps requires different techniques to execute the expected tasks; all of these techniques are discussed in the following section, dedicated to the state-of-the-art techniques.

2.2.1.3. State-of-the-art of CBRSs

Research on CBRSs takes place at the intersection of Information Retrieval (IR) and Artificial Intelligence (AI). Since users in RSs search for relevant recommendations, are engaged in an information seeking process. Hence, recommender systems can be seen as IR systems where the user's profile acts as a query (i.e. A permanent filter expressing the long term user's preferences instead of a set of keywords). When it comes to Artificial Intelligence, the recommendation process can be seen as a learning problem that capitalizes on past knowledge about users. This requires the use of Machine Learning (ML) techniques to learn a personalized user's profile expressing his past preferences.

The following sub-sections introduce the state-of-the-art of techniques used during the CB recommendation process, which covers the three main steps already discussed:

a. Item representation

Generally, the descriptions of items are textual features, i.e. not features with well-defined values. In fact, CB systems were developed to filter and recommend text-based items (e.g. Documents, web pages, news articles, product description, etc.) based on a list of relevant keywords present within the system's documents. Hence, the content of documents can be represented as a set of terms or keywords that it contains (called also features, attributes or properties).

Most content-based recommenders use relatively simple retrieval models, such as keyword matching or the Vector Space Model (VSM) with basic Term Frequency Inverse Document Frequency (TF-IDF) weighting. VSM is a spatial representation of text documents, in which the TF-IDF vectors are large and very sparse, so to make them more compact and eliminate irrelevant data, many techniques can be applied such as: removing stop words and stemming [22], removing size cutoff [23, 24] or using phrases instead of terms [25].

In addition, Textual features create a number of complications when learning a user profile, due to the natural language ambiguity (synonymy and polysemy). To solve these issues, semantic analysis (lexicon, ontologies [26], Encyclopedic knowledge [27, 28]) are introduced, in order to annotate items as user's profiles. This can help to have a semantic interpretation of user's needs.

b. Learning user's profile

In the task of building the user's profile, many Machine Learning techniques are used, which are adapted for text-categorization context [29]. The process of learning profiles consists of building text classifiers by learning the features of categories, based on ***training set***. This last, contains documents which are labeled with the category they belong to. Since each document can be either interesting for the active user or not, we can distinguish two types of categories:

- C+: documents liked by the user
- C-: documents disliked by the user.

The techniques used in the context of content-based systems require that the user assign a relevance score to documents, to be labeled later. This can help thereafter to predict the relevance of a document for a known user. Different techniques can be exploited by CBF to model the relationships between different textual items within a dataset [19], These techniques make recommendations by learning the underlying model with either: 1) statistical analysis like K-Nearest-Neighbors (KNN) and Relevance feedback – Rocchio's method or 2) machine learning techniques as Probabilistic models [23, 30], Naive Bayes Classifier, Decision Trees, Decision rule classifiers, or Neural Networks [8, 31, 32].

In [23] a comparative evaluation covered the learning algorithms have shown that decision trees and K-NN don't give accurate results, on the contrary of The Bayesian and Rocchio's methods which perform well in all domains.

c. Generating recommendations

The CB approach does not base its inference on ratings of other users as the case in CF approaches, Instead it relies on the description of the items. The key component of such approach is the similarity function that computes the distance between items in n dimensional space (the similarity and distance are inversely proportional, i.e. when the distance is large between two items they are less similar and vise-versa).

CB similarity focuses on an objective distance among the items, without introducing any subjective factor into the metric (as CF approach). Most of the distance metrics deal with numeric attributes, or single feature vectors. Some common distances, given two features vectors x and y, are: Euclidean, Manhattan, Chebychev, cosine distance for vectors and Mahalanobis distance [5, 33].

2.2.2. Collaborative Filtering (CF)

2.2.2.1. General approach

The Collaborative Filtering (CF) technique is considered as the most prominent approach to generate personalized recommendations, since it is widely used by many commercial e-commerce sites in different domains (books, movies, jokes, etc). This approach uses the "wisdom of crowd" to infer recommendations, its underlying assumption is that if certain users shared the same tastes and preferences in the past, they tend to choose similar items in the future. It extends the concept of "word of mouth" among friends or other similar people on the internet (i.e. Some friends or thousands of people are likely to recommend what they prefer or to give their opinion about the products they consumed). The rationale is to filter items that are likely appreciated by the user from a large set of items, relying on preferences of similar users (Hdioud et al. 2012). The reason behind calling the technique "*collaborative filtering*" is that users collaborate with each other to make good choices concerning items, it's called also *community-based*.

CF approaches [34, 35], don't exploit or require any knowledge about the items themselves (e.g. Genre or author of the book) like content-based approaches, but rather they rely on the ratings of the active user as well as those of the others available in the system to infer similarity between users or items. The only required input for such approaches is the user-item ratings matrix, which is exploited to generate recommendations under two forms: 1) Best item with a numerical rating prediction that indicates to what extent the user will like or dislike the item in question, or 2) *Top-N items* as a ranked list of items sorted in descending order according to their computed prediction value (see Figure 8).

Figure 8: Pure CF approaches: inputs and outputs.

The advantage of this strategy is that these data do not have to be entered into the system or maintained. Furthermore, recommended items are various, novel and not with the same content as recommendations offered by content-based recommender (over-specialization issue). Generally, the CF methods are often classified as being either *Neighborhood-based* or *Model-based* (see Figure 9):

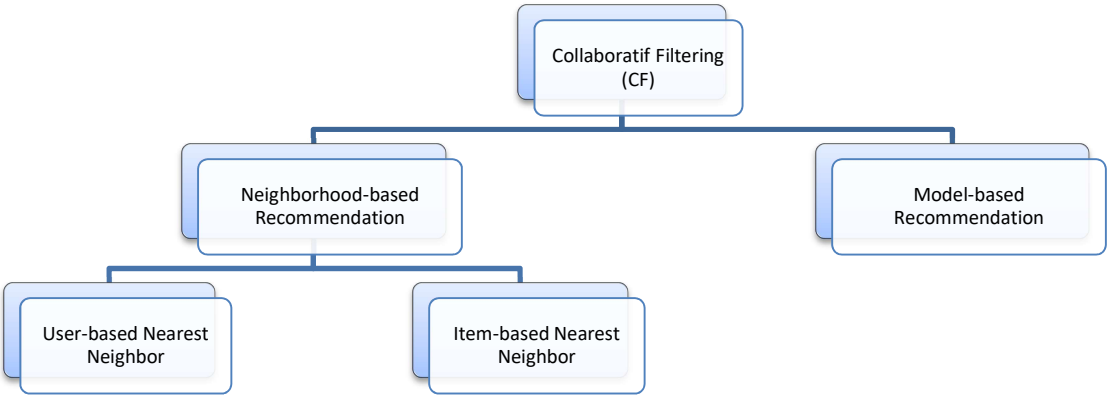


Figure 9: Classes of CF approaches.

- **Neighborhood-based approaches:** rely on the opinion of the like-minded people to the active user. It uses directly the user-item matrix, we distinguish methods based either on similar users or on similar items (Hdioud et al. 2012):

1) *User-based:* based on ratings of the active user and his neighbors which are users with similar tastes (rate the same items in almost a similar way).
2) *Item-based*: based on ratings of the active user on items that are similar to a given item (similar items are those rated by several users in a similar fashion).

- **Model-based approaches:** use the dataset of ratings to learn a predictive model by extracting some information. This model consists of a user's presentation describing his preferences as well as a category class of items that will be used to predict the rating of new item. In other words, the "model" is used to make recommendations without having to use the complete dataset each time the recommendations have to be computed, which offers a high speed and scalability.

The main difference between these two techniques lies in the ability of each to scale as the number of users and items in the system grows which is the case in real applications (we talk about tens of millions of users and items). Neighborhood-based approaches are also said to be *Memory-based* [5, 36], because they act on the whole original ratings matrix memorized in database that is used directly to generate recommendations. Even if these approaches are more precise since full data is available and used, but they face a real problem of scalability. The exact algorithm is intrinsically quadratic: the time to build the model is proportional to the square of the number of objects to compare. On the other hand, Model-based approaches process the raw data in offline mode. Then, at run time (online mode) only the pre-computed learned model is used to make necessary computation to make recommendations, hence the decrease of response time.

2.2.2.2. Neighborhood-based approaches

Neighborhood-based, K-Nearest-Neighbor (KNN) or Memory-based are different nominations that designate one single approach which is a mainstream in RS field. Such methods are very popular; they were used from the onset of CF recommenders [35] and until now they are extensively used either in industrial context or as a combination of other methods [37, 38, 39]. All KNN methods for CF require common components:

- Similarity weight function: that computes the degree of similarity for each pair of objects (users or items).
- Neighborhood selection: based on the similarity function, giving for each object its list of most similar objects (items or users),
- A method for combining the ratings and similarities to generate ratings predictions and recommendations for unseen items.

a. User-based Nearest Neighbor recommendation

The user-based approaches compute the rating prediction $\widehat{r_{ai}}$ for a new unseen item *i*, by using the ratings given to the item i by the nearest-neighbors of the active user *a*. Hence, methods to compute similarities between different users, as well as to select the most relevant user's neighbors are required (more details are presented in subsections *c* and *d*).

We denote N_a by the set of nearest neighbors of the active user (with $|N_a| = K$ users) and I_u the set of items rated by the user u. The predicted rating of the active user *a* on unrated item *i* is the weighted sum of the ratings of similar user that have already rated the item *i*:

$$\widehat{r_{ai}} = \frac{\sum_{(u \in N_a | \, i \in I_u)} sim_{au} \cdot r_{ui}}{\sum_{(u \in N_a | \, i \in I_u)} sim_{au}} \qquad \text{(eq.1)}$$

However, this equation represents a flaw; since it doesn't take into account the difference in the use of the rating scale by different users (i.e. It considers that all users express their satisfaction against an item with the same manner). For instance, in a 1-to-5 rating scale, two different users can give a value 5 to a given item, if the first has a habit to give a high rating for all items when the second user is not easy to please and all his ratings are low and this time he gives a 5 as a rating's value that mean he is very satisfied by this item especially. Hence, the ratings' values are the same, but don't express the same level of satisfaction for the two different users. The solution is normalizing the ratings to convert a personal rating into a universal scale, predictions based on deviations from the mean ratings have been proposed. In that case, $\widehat{r_{ai}}$ is computed using the sum of the user mean rating and the weighted sum of deviations from their mean rating of the neighbors that have rated the item i:

$$\widehat{r_{ai}} = \overline{r_a} + \frac{\sum_{(u \in N_a | \, i \in I_u)} sim_{au} \cdot (r_{ui} - \overline{r_u})}{\sum_{(u \in N_a | \, i \in I_u)} sim_{au}} \qquad \text{(eq.2)}$$

Where $\overline{r_u}$ is the mean rating of the user *u*.

The time complexity of user-based approaches is O(m^2 × n × K) for the neighborhood model construction, it is O(K) for one rating prediction, and the space complexity is O(m × K), with *m* the number of users and *n* the number of items.

b. Item-based Nearest Neighbor recommendation

The predicted rating's value is computed according to the same principle, the only difference is that instead of relying on the opinions of the like-minded people, we use the rating of similar items to the concerned item for which we want to predict the rating. We note N_i the neighborhood of the item *i* (with $|N_i| = K$ items). Symmetrically to the user-based approach; the predicted rating that the user *u* would give to the item *i* is computed according to two ways is as follows:

The weighted sum:

$$\widehat{r_{ai}} = \frac{\sum_{(j \in N_i \cap I_a)} sim_{ij} . r_{aj}}{\sum_{(j \in N_i \cap I_a)} sim_{ij}} \quad \text{(eq.3)}$$

The weighted sum of deviation from the mean of the ratings:

$$\widehat{r_{ai}} = \overline{r_i} + \frac{\sum_{(j \in N_i \cap I_a)} sim_{ij} . (r_{aj} - \overline{r_j})}{\sum_{(j \in N_i \cap I_a)} sim_{ij}} \quad \text{(eq.4)}$$

The time complexity of item-based approaches is O(n2 × m × K) for the neighborhood model construction, it is O(K) for one rating prediction, and the space complexity is O(n × K), with *m* the number of users and *n* the number of items.

c. Different similarity metrics

The similarity measurement quantifies the degree of similarity between two objects (either users or items). It plays a crucial role in RSs rested on neighborhood-based techniques, as they allow selecting the best neighbors to be used in prediction step with different degrees of importance (weighting). The similarity has a significant effect on the accuracy of recommendations and overall RS's performance. In fact, there is no single definition of a similarity, but generally similarity measures are the inverse of distance metrics (i.e. The more the distance value is small, the most similar objects we have and vice versa). The most important means to measure similarity between users or items are those based on correlation such as Cosine and Pearson coefficients which are the first proposed by Resnick et al. [35]. Other similarities, such as the adjusted cosine, distance of Manhattan, Jaccard can be used. The similarity between two users u and v is computed as follows:

$$Cosine(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} \, r_{vi}}{\sqrt{\sum_{i \in I_u} {r_{ui}}^2 \sum_{i \in I_v} {r_{vj}}^2}} \quad \text{(eq.5)}$$

$$Pearson(i, j) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})(r_{vj} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})^2 \sum_{j \in I_{uv}} (r_{uj} - \overline{r_v})^2}} \quad \text{(eq.6)}$$

The difference between the two measures is that the cosine similarity doesn't take into account the differences in the mean and variance of the ratings given by the items u and v, in the contrary to the Pearson measure which eliminates the effects of mean and variance when comparing the ratings of two users.

d. Neighborhood construction

Generally, it's impossible to include all the neighbors during the calculation of predictions as it will affect the RS's performance (i.e. High computation time). Furthermore, the use of a large number of nearest neighbors whose similarity weights aren't trustworthy sufficiently influences badly the recommendations'

accuracy and coverage [40]. Subsequently, it is necessary to select very carefully high-confidence neighbors. There are two common techniques used in this regard to reduce the size of the neighborhood to be used in prediction step: 1) Top-N filtering: limit the size of the neighborhood to a fixed value (for each user or item only N neighbors are kept and taken into account) or 2) Threshold w_{min}: defining a specific minimum threshold of similarity.

In fact, the choice of the two parameters N and w_{min} is very crucial and influences the obtained results in the future. For instance, if the threshold is high, only small neighbors are selected, then it will be impossible to compute prediction for a large number of items (a small or limited coverage). On the other hand, when the threshold is very small, a big neighborhood is constructed, but subsequently the computed predictions are not accurate (low accuracy). For more details about the effects of using different similarity metrics as well as neighborhood construction on the overall accuracy and performance is reported in [41] by Herlocker et al. (2002).

e. User-based Vs. Item-based

There are five criteria that should be taken into account, when it comes to choose between the two approaches (Table 2):

1) **Accuracy:** at this point the choice between the two techniques is not final; it depends mainly on the ratio between the number of users and that of items in the system. In the case where the number of items is much greater than that of users (e.g. Research papers recommender), the user-based approach can perform more accurately [42]. Similarly, when the number of users exceeds largely the number of items, like in online store and e-commerce applications with thousands of hundreds of users but only many hundreds of items to be sold (e.g. *amazon.com*); item-based approach is much favored [43, 44].
2) **Efficiency:** the memory and computational efficiency of RSs also depends on the ratio between the number of users and items. Thus, when the number of users exceeds the number of items, as is it most often the case, item-based recommendation approaches require much less memory and time to compute the similarity weights (training phase) than user-based ones, making them more scalable. However, the time complexity of the online recommendation phase, which depends only on the number of available items and the maximum number of neighbors, is the same for user-based and item-based methods.
   In practice, computing the similarity weights is much less expensive, due to the fact that users rate only a few of the available items. Accordingly, only the non-zero similarity weights need to be stored, which is often much less than the number of user pairs. This number can be further reduced by storing for each user only the top *N* weights, where *N* is a parameter. In the same manner, the non-zero weights can be computed efficiently without having to test each pair of users or items, which makes neighborhood methods scalable to very large systems.
3) **Stability:** it depends on the frequency and amount of change in the users and items on the system. If the list of available items is fairly static in comparison to the users of the system, an item-based method may be preferable since the item similarity weights could then be computed at infrequent time intervals while still being able to recommend items to new users. On the contrary, in applications where the list of available items is constantly changing, e.g., an online article recommender, user-based methods could prove to be more stable.
4) **Justifiability:** at this point, item-based techniques have a major advantage, since they can be used easily to justify the afforded recommendations, e.g. the list of neighbor items used in the prediction, as well as their similarity weights, can be presented to the user as an explanation of the recommendation. By modifying the list of neighbors and/or their weights, it then becomes possible for the user to participate interactively in the recommendation process. Unlike user-based methods which are less amenable to this process because the active user does not know the other users serving as neighbors in the recommendation.

5) **Serendipity:** In item-based methods, the rating predicted for an item is based on the ratings given to similar items. Consequently, recommender systems using this approach tend to recommend to a user items that are related to those usually appreciated by him. While this may lead to safe recommendations, it does less to help the user discover different types of items that he might like as much. (It's almost the same thing as content-based RSs).
On the other hand, user-based approaches are more likely to make serendipitous recommendations. This is particularly true if the recommendation is made with a small number of nearest neighbors.

Table 2: Comparison of user-based and item-based according to five criteria.

	Accuracy	***Efficiency***	***Stability***	***Justifiability***	***Serendipity***
User-based	|U|<|I|	|U|<|I|	Items' list in continuous change	Hard explanation	Various recommendations
Item-based	|U|>|I|	|U|>|I|	Available items are static with few changes in time	Easy explanation	Similar recommendations

The user-based techniques were widely used and very successful at a certain moment. According to [37], Deshpande and Karypis, 2004 models based on user-user matrices give a better predictive performance than those based on item-item matrix. On the contrary to [38] Sarwar et al., 2001 which favored item-based techniques which have replaced gradually user-based approaches latterly [34, 37, 45], due to its revealed related issues: sparsity and scalability [38]. In fact, the pure KNN algorithm is a quadratic function which requires an increased computation and consumes much more memory, as the number of users and items increases. In real world applications, the number of users often goes far beyond the number of items. Hence, it is wise to opt for the use of item-based approach. However, this advantage would be less important today as many user-generated catalogs (user-generated video on Youtube,...) or recommender applications to the web or to the music can lead to very huge catalogs with more items than users.

Table 3: Comparison of user-based and item-based at the functional level.

	Predictive Performance	***Easy explanation***	***Management of ratings***
User-based	✓		
Item-based		✓	✓

There are many works that compared user-based and item-based techniques [34, 39, 46, 47] from a functional point of view. The superiority of item-based doesn't lie only on the predictive performance, but also there are two other main reasons for which it excels over the others (see Table 3):
· **Easy explanation:** recommendations offered by this type of method are very easy to explain to users, which ensures a transparency between the RS and its users (e.g. If a user *u* gave a good rating to an item *i* and item *j* is similar to item *i*, subsequently the item *j* will be recommended).
The so-called item-to-item recommendation is adopted by many industrial systems. For instance, Amazon[1], which uses the famous archetype associated message "people who have seen/bought this item also viewed/purchased these items".
· **Easy management of the new ratings:** some new ratings of a user do not significantly modify the item-item similarity matrix. Therefore, it is not necessary to recalculate the matrix when a user rates a few additional items. As at the same time the prediction formula based on the item-item similarity matrix

[1] www.amazon.com

takes into account the immediate change of a user profile, it is very easy to manage new users (with one rating) or user's profile change.

2.2.2.3. Model-based approaches

In real cases, Memory-based RSs suffers often from scalability issues, especially when they are used in the context of real industrial recommenders, where real-time recommendations are generated using very massive data. To overcome these difficulties the model-based techniques are employed, in which the raw data are first processed offline or some *dimensionality reduction* techniques are applied. At the time of execution, only the pre-calculated data is in use.

In fact, there are a number of techniques that can be taken to build a model and use it to compute predictions. Among these techniques, we find the most traditional KNN algorithm, which can be enhanced during the processing. The operating principles still the same concerning the computation of similarities and using them as "weights" to predict a rating for an unrated item. However, similarities (among either users or items) are calculated offline then they are stored as a model. Yet, in real cases models are built using similarities between items rather than users which is desirable since the number of users often exceeds largely the number of items (e.g. The NetflixPrize data contain slightly fewer than 5,00,000 users, but only a little over 17,000 movies). In addition, to achieve a high level of scalability only a limited number K of similar neighbors (users or items) can be stored and used later [38]. Further techniques used in Model-based Recommendations, the most important and known approaches are:

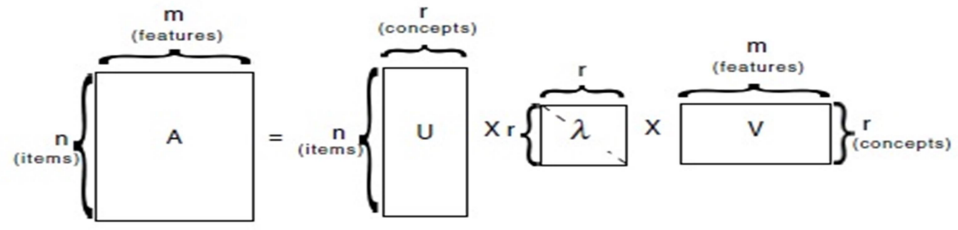
a. Matrix Factorization techniques and Latent Factor models

Matrix Factorization (MF) techniques are included in dimensionality reduction methods whose aim is presenting both of items and users in a compact space, i.e. more reduced, which capture more salient features, this last is called *latent space*. Since users and items are represented in this dense space a significant relation can be revealed between users even if they haven't rated the same items, or haven't rated sufficient numbers of items, which solve the problems of *Sparsity* and *limited coverage*.

In fact, the use of MF techniques in the context of RSs became popular during the Netflix challenge [48] since they are more accurate and speedy. These techniques are extensively used to improve RSs based on KNN approaches, by decomposing a given matrix M- generally ratings matrix or similarity matrix-into three simpler ones. By doing so, a set of latent (hidden) factors are derived from the original matrix, then both of users and items are represented as vectors of factors. To do so, Latent Semantic Analysis (LSA) techniques- also referred to as Latent Semantic Indexing (LSI) - that relies on Singular Value Decomposition (SVD) are applied:

The general formalization of SVD is as follows:

Given a $n \times m$ matrix A with the rank r, the SVD of A is defined as: $SVD(A) = U \lambda V^T$.



(eq.7)

Where:

- Columns of U are the eigenvectors of AA^T.
- Columns of V are the eigenvectors of A^TA.

- The matrix λ is a diagonal matrix that contains singular values; these last are the square root of the eigenvalues of AA^T and A^TA both.

The SVD of R gives the best linear approximation of A when selecting the k first columns of U, the k first singular values of λ, and the k first rows of V. The interest of applying SVD lie on the fact that it gives the approximation of k-factor-based $U \times\lambda\times V$, with $k<< m$ and $k<<n$, which makes an important compression of information with a data denoising.

Concerning the LSI/ LSA method, it permits to establish relations between items and features (e.g. Terms and documents), to construct concepts that rely on both items and features. To do so, the matrix of occurrences (rows are terms and columns are documents) is transformed into relations between items-concepts and concepts-features. To this end, we reduce the rank of the matrix n into a rank k (where $k<n$) which is equivalent to the decomposition of the matrix of occurrences with SVD, to find concepts space:

$$\begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{pmatrix} = \left(\left(u_1\right)...\left(u_l\right)\right).\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_l \end{pmatrix}.\begin{pmatrix} \left(v_1\right) \\ ... \\ ... \\ \left(v_l\right) \end{pmatrix} \quad \text{(eq.8)}$$

Where:

- The component that contributes in the *i-th* item is the *i-th* row of U.
- The component that contributes in the *j-th* item is the *j-th* column of V^T.

Since the items and their features are represented in concepts space, to compute the correlation between components we use the new components.

b. Clustering

The idea of the clustering model is to build groups (clusters) of either person with similar tastes, or items that have the same subjects, or that tend to please the same people. Thus, to predict the rating that a given user will give to an article, the opinions of users who belong to the group are used. In other words, we want to associate a class to each user, as well as individual items. But these classes are a priori unknown, so they must be derived from the model estimation process. The clustering techniques allow limiting the number of objects (users or items) considered in the calculation of the prediction. The processing time is decreased and the results are potentially more relevant since the observations concern a group of users with similar behavior (items with the same characteristics).

Typically, clustering systems differ in the *objective function* chosen to evaluate the quality of clustering, and the *control strategy* for space travel of possible clusters. But they all follow the general principle of traditional clustering that is maximizing the similarity of observations within a cluster (intra-class), and minimizing the similarity between clusters of observations (extra-class), to obtain a partition of the base as relevant as possible.

K-means:

This algorithm is the best known and most used, due to its simplicity of implementation. The K-means algorithm is divided into four steps, as follows:

1. Randomly choose *K* objects to form the *K* initial clusters. These objects are centers of clusters that contain only one element at the first time.
2. Reassign objects in a cluster. Each object *x* is affected to the class which he is closest to the center, according to a distance measure: Pearson, Cosine, etc...
3. Recalculate the new centers of *K* clusters.
4. Repeat steps 2 and 3 until no more reassignment is possible.

The K-means method is used in many RSs. However, this method has some flaws related to the choice of initial clusters; also it is very difficult to know in advance the number *k* of suitable centers which affects the quality of classification. On the other hand, this algorithm is expensive in computation time and can present convergence problems. Indeed, the generated clusters are highly dependent on the initialization phase of the algorithm and it is often rare to come across an overall optimum that minimizes intra-group distances and maximizes inter-group distances. In general, it is common for the algorithm not to end. Moreover, the results are non-reproducible, that is, if the algorithm is started twice with identical parameters, the results are likely to be different. Despite the popularity of K-means there are many competing algorithms such as Repeated Clustering or Hierarchical clustering.

Repeated Clustering:

The idea here is to group users and items separately. At the first phase, users are grouped into clusters based on the items, and items are grouped into clusters based on users, and in a second stage, individuals are grouped into clusters based on clusters of items, and articles in terms of clusters of users. These two phases are repeated until satisfaction of the clustering obtained, or a fixed number of times [49].

Hierarchical Clustering (RecTree):

Hierarchical clustering is based on constructing a *tree of clusters*. We start by decomposing users or items in two clusters, using the nearest neighbor algorithm to maximize similarities within the cluster and minimize outside cluster. The operation is repeated for each cluster obtained. The figure 10 represents a tree of clusters; where the leaves of this tree represent the clusters to be used in the prediction of the evaluation [50].

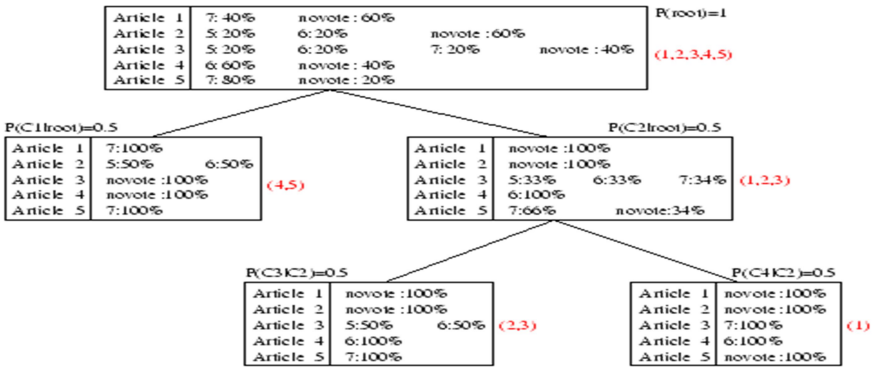


Figure 10: Hierarchical Clustering algorithm.

c. Bayesian network and probabilistic model

Bayesian classifiers [51, 52] consider all features and classes of a learning problem as random continuous or discrete variables. Using conditional probabilities and the Bayes' Theorem, the goal of a Bayesian classifier is to maximize the posterior probability of the class of any item to classify given data. Using the ratings as classes with discrete values, a Bayesian classifier can be applied to real-valued rating data.

The Bayesian network is presented as a directed acyclic graph that represents a probability distribution of dependence between a set of entities (users or items). Each node in the graph represents an entity, and each arc a direct dependence between variables. Thus, each variable is independent of its non-descendants in the graph, given the state of his parents (see Figure 11).

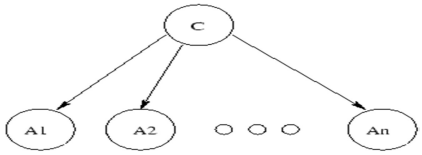


Figure 11: Bayesian Network**.**

The algorithm is learnt based on a training set, by searching among the different possible model structures, in terms of dependence. So, in the resulting network each item may have a set of resources that relatives are the best predictors of his votes. In the context of collaborative filtering RSs, the rationale is to associate a Bayesian network to each base item. Each leaf of the tree is a probability of a rating for the article, given the state of the parents identified. To predict the possible rating of a customer for an item, we move in the corresponding Bayesian network, according to ratings that the user gave to *parent's items* present in the network, and then the most likely score is attributed to the Article considered.

Further data mining techniques can be used in the context of CF to learn a model, such as decision trees and association rules [53, 54].

2.2.2.4. Discussion

Recent surveys have shown that model-based approaches are superior to those based on neighborhood in term of prediction accuracy, but this last doesn't ensure the user's satisfaction. Another factor that has a big influence in the appreciation of the user is serendipity. In fact, there is a large difference between *novelty* and *serendipity*, the former guarantees that the system offer to its users novel recommendations, that they likely weren't aware of them, but that it wasn't possible to find them, unlike the latter which presents to the user original items that are difficult to find.

The neighborhood-based systems include the serendipity factor, because the recommendation is based on local similarity between users and items. Thereafter, the system recommends the item that the users' neighbors have liked, this item may be not appreciated or novel for the user, but it helps the user to discover different things, which he will not have the occasion to find somewhere elsewhere.
The main advantages of neighborhood-based methods are:
• Simplicity: this type of systems is intuitive and easy to implement, doesn't require much parameters to be tuned, just the similarity metric used and the number of neighbors.
• Justifiability: when a system offers recommendation based on this type, it may include also a concise justification for the computed prediction to the user that represents on which criteria the system has generated such recommendation, which ensures a transparency between the user and the RS as well as it improves user's confidence into the system and his recommendations.
• Efficiency & stability: unlike the model-based, there are no training or learning steps which cost very much in term of computing. Neighborhood methods use directly the ratings and compute similarity in the off-line step. For the stability, there are no serious changes in the addition of the new members, in the contrary of the model-based which requires re-training of the whole system in new coming members.

On the other hand, one point that is often overlooked is that model-based techniques wouldn't possibly achieve the prediction accuracy expected compared with memory-based approach. This is due to the fact that not all the available data is used when predictions are computed. But generally, the prediction quality depends on how the model is built. In addition, building models is a time and resource consuming process, which makes them inflexible when it comes to add more data to the model. Thus, many researchers are

investigating their effort in studying and enhancing model-based CF. However, model-based approaches have better scalability and generate predictions with high-speed unlike memory-based.

2.2.3. Hybrid

2.2.3.1. General approach

Each of the stand-alone recommendation approaches has its pros and cons. The main goal of hybrid recommendation is to compensate the weaknesses of each one and combine their strengths, in order to achieve better accuracy and high-level performance. For instance, the existing complementarity between CF and CB approaches makes them good candidates for hybridization, since CB is essential to bootstrap RS when CF lack sufficient data which helps CF overcome the cold-start problem. On the other hand, CF supports CB to avoid the overspecialization issue as CF offer serendipitous recommendations. The Hybrid approaches consist in combining two or several recommendation paradigms into one single RS, and generally they are based on different data sources in different natures and forms.

2.2.3.2. Burke's classification

In fact, there are two dimensions that must be taken into account when it comes to hybridize two recommendation techniques or more: 1) *recommendation paradigms* to be hybridized, which consist in defining the recommendation components on which the future hybrid RS will be based 2) *hybridization design* that designates how different recommendation algorithms are integrated into one hybrid RS.

Therefore, the choice of which recommendation paradigms are used defines the type of required input data, such as ratings, item's features, explicit knowledge domain, demographic data, etc. On the other hand, the combination of different approaches into a hybrid one can be realized according to many strategies. Burke (Burke, 2002) and (Burke, 2007) presents his taxonomy in which he distinguishes between seven types of hybridization that can be abstracted into three major designs (see Figure 12):

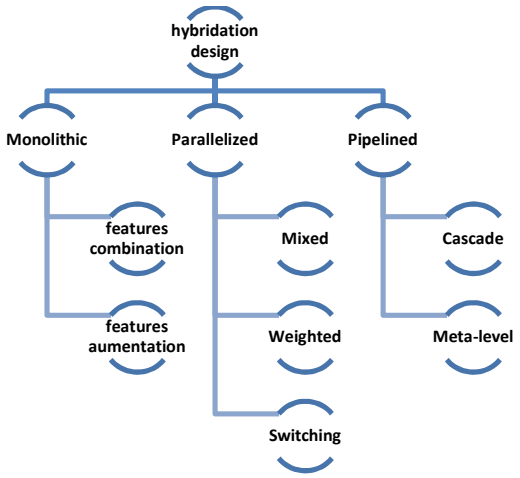


Figure 12: Hybridization designs according to Burke's Taxonomy (Burke 2002, 2007).

1) **Monolithic:** designates a hybridization design that incorporates aspects of several recommendation strategies in one algorithm; it consists in a single recommender component integrating multiple approaches by processing and combining them, on the contrary to the two other existing hybridization designs which require at least two separate recommendation components to be combined. However, in this case the different recommender components contribute *virtually* in the hybrid recommender, because that the hybridization is performed by integrating a modification of algorithm behavior to exploit different types of inputs data, which

are either: 1) Specific by other recommendation algorithms and used by the hybrid recommender (features combination) or 2) Augmented by one technique and exploited by others (features augmentation).

- *Features combination:* It consists in using a diverse range of input data, which are merged later in order to improve accuracy. A real application of this strategy was reported in [55], which consists of combining collaborative information (e.g. Liked / disliked) and item's features derived from content-based techniques, which identify new hybrid features based on community and product data.
  Another approach for feature combination was proposed in [56], who exploit different types of rating feedback based on their predictive accuracy and availability. In fact, Burke said, that the principle of "Feature combination" is not a hybrid method in the strict sense as there is no use of several recommendation systems.
- *Features augmentation:* It is used to integrate several recommendation algorithms; his task is not limited only on combining simply several types of input as the features combination, but also it applies a complex transformation which leads to augmenting the feature space. Content-boosted collaborative filtering is an actual example of this variant [57]. It predicts a user's assumed rating based on a collaborative mechanism that includes content-based predictions.

2) **Parallelized:** In contrast to the previous design, this one requires two recommender implementations or more that operate independently the one of the other, these implementations are employed side by side, as their inputs are aggregated to derive recommendations. This design consists in three different strategies:

- *Weighted:* It consists in combining the separate results of each recommender. The output in this case is a scoring of each recommended item obtained by taking the weighted average scores of each recommender for this item. In a more formalized way the result set is as follows:

$$rec_{weighted}(u,i) = \bigcup_{k=1}^{n} \beta_k \times rec_k(u,i) \text{ Where: } \sum_{k=1}^{n} \beta_k = 1 \quad \text{(eq.9)}$$

- *Mixed:* This strategy allows combining all outputs (recommended items) derived from different recommender systems in the level of the user interface; it seems practical when a large list of recommendations must be displayed to the user. Thus the result set is a set of tuple <score,k> as follows:

$$rec_{mixed}(u,i) = \bigcup_{k=1}^{n} \langle rec_k(u,i), k \rangle \quad \text{(eq.10)}$$

- *Switching:* This strategy as its name indicates, allows switching between the different recommender techniques depending on the situation:

$$\exists k : 1 \ldots n \; rec_{switching}(u,i) = rec_k(u,i) \quad \text{(eq.11)}$$

  Where k is determined by the switching condition. For example, this strategy can be used to overcome the problem of cold-start in the collaborative approaches, to this end we can switch between Knowledge-based and collaborative techniques. The former is used initially to recommend items even those new added, until sufficient ratings are available then the later can be used.

3) **Pipelined:** consists in implementing a recommendation process with different *stages* in which several recommendation techniques are carried out in *sequential* manner. The pipelined hybrid variants differentiate themselves mainly according to the type of output they produce for the next stage in such a way that the outputs of one recommender are inputs for the next one. This design consists in two different strategies:

   - *Cascade:* This technique is based on a sequenced order of recommender techniques, where the first one is used to produce a coarse ranking of items and the succeeding recommender refine the recommendations. For instance, EntreeC [20] is a cascaded knowledge-based and collaborative recommender that uses its knowledge about restaurants to make recommendations based on the user's tastes.
   - *Meta-level:* This strategy consists of combining two recommenders in such way that the RS exploits a model (as input) generated by its predecessor.

2.2.3.3. Discussion

To summarize, all basic recommendation techniques can be enhanced by being hybridize with each other. But generally, there is no specific hybridization design that can be applied in all circumstances or a general rule to identify which variant is more suitable. It depends only on our needs and aims to alleviate their shortcomings and benefit from their advantages. On the other hand, despite the several benefits of hybrid systems, but they have some possible limitations that consist on the increased complexity of RS. Moreover, the combination of many recommendation paradigms according to some complex designs can have a bad influence on speed, since more models are executed simultaneously.

2.3. Other Recommendation approaches

Even recommendation techniques are broadly categorized into three main approaches that were discussed in details in the previous section. Although, it is worth mentioning that there exist further approaches such as Knowledge-based and demographic Filtering-based, those are also considered among the general recommendation approaches according to (Burke 2002). However, these techniques receive poor attention and are seldom used in industrial recommendations.

Furthermore, recently several new emerging techniques have begun to capture more attention, which exploit user-oriented approaches in RSs instead of classical Machine Learning (ML) techniques. Those techniques have for aim generating more relevant and personalized recommendations by getting the user involved during the recommendation process. To this end, contextual and psychological (emotions and personality) aspects are investigated.

This section aims to introduce these different approaches briefly and gives an insight about them.

2.3.1. Knowledge-Based (KB)

The basic traditional recommendation approaches – CB and CF– are more suited to recommend items like books, movies, news, etc, but aren't valid for other application domains as *Electronics consumer* in which we find a large number of ***one-time*** buyers for items like cars, phones, computers, etc, that aren't bought often. As a consequence, there is no need to save a "*purchase history*" [58, 59]. In addition, in such domains the ***time span*** plays a crucial role for recommendations that cannot be based on years-old preferences due to the change of user's preferences over time because of their lifestyles and fluctuation in such markets. Another recommendation approach has been appeared that meets with these requirements is Knowledge-Based (KB) which was addressed in [60].

Such recommendation approach relies on the explicit users' requirements and deep knowledge about the application domain. The recommendation process according to KB recommenders is carried out as follows:

1) The user expresses his preferences in a detailed way, i.e. he specifies the items' features and characteristics which interest him.
2) The KB recommender tries to find out items based on the specific domain knowledge by matching up items' features to meet with users' requirements.
3) Reasoning out items that fit with the preferences expressed.

In fact, KB approaches have many advantages compared to the traditional ones. Since the users express explicitly their preferences in a detailed way, once they need recommendations, there is no need to maintain users' profiles. As a result, there is no cold-start problem in such context. In addition, such recommenders ensure a high interactivity with the user, i.e. instead of relying on filtering out items which likely interest the user, the recommender guide the customer in a personalized and interactive way to most useful items.

There are two specific types of KB recommenders which are both similar in term of the use of knowledge (detailed knowledge about items), but they differ in the way on how recommendations are computed:

- Constraints-based: in this approach the system exploit predefined recommender knowledge bases relying on explicit recommendation rules, that connect customer requirements with item's features. In order to recommend a given item, it must fulfill these rules.

- Case-based: based on similarity metrics to compute to what extent given items match with customer's preferences.

2.3.2. Demographic-based (DM)

In [61] Pazzani investigated the Demographic Filtering approach, it attempts to form "people-to-people" correlations like CF but uses different data. This type of RS is a special case of community-based recommendation, where users are categorized into classes based on their similar personal data such as (age, gender, education, etc) instead of ratings as is the case of CF approach. Hence, such approaches require a prior knowledge about demographic information about users to be able to generate recommendations [12]. These systems are stereotypical (i.e. Create "stereotypes") since they depend on the assumption that users that have the same demographic features tends to have similar preferences concerning items.

A "stereotype" is a collection of frequently occurring characteristics of users [12]. For instance, one might guess that if a given customer is a teenager, he might prefer watching adventure or horror, since all teenagers like these kinds of movies. Moreover, if he concerned customer is a teenager girl, she will definitely enjoy watching a romantic movie and so on.

One of the earlier demographic-based recommenders, we mention Grundy system [62], which is one of the first books' RSs that recommends books based on users' personal data acquired by an interactive dialogue with them. Another example in this category is the LifeStyle Finder [63], which suggests items and web pages to users.

Advantages and disadvantages

The demographic approach has many advantages when it is compared to the others. One of its main strength resides in the fact that it doesn't require the history of user's feedback like CF and CB approaches. Hence, new users may benefit from recommendations, even if they haven't rated any item yet. In addition, this technique is domain independent since any knowledge about items' features is needed. Also, demographic approach is very easy and simple to be implemented and generates recommendations quickly based on a limited amount of information.

On the other side, this technique has some minuses represented in: first, the difficulties related to gathering demographic information due to privacy issues since some users could considerate it as personal data. Second, recommendations inferred by such technique are crude and very general, while users look for more personalized recommendations. Further issues from which demographic-based approach suffers are discussed in section (2.4).

2.3.3. Context-aware

As we had already seen in the previous sections, all recommendation techniques focus on recommending relevant items to users, by considering only two main entities (user and item), without taking into account any additional information about time, location, company, etc. However, in some situations putting the user or an item into a well-defined context may be very helpful to generate relevant recommendations. In fact, in many application domains the usefulness of certain products to a user depends heavily on time (e.g. Season, evening, night) and once it depends on the people with whom the product will be used and under what circumstances. For instance, recommending a movie depends heavily on the person with whom the user plans to see the movie, regardless his own preferences: if the user will watch the movie with his *children,* he will prefer to a *cartoon* movie, but if he is with his *conjoint* he will enjoy a romantic *movie*). The time also may have an effect, as in the holidays he might prefer films in relation to this period.

In RSs domain, we denote the context as being a set of situation parameters that influences the selection and the ranking of recommendation results. In [64], Shilit et al. (1994) define the most important aspects of context, such as the location where you are, who is with you and what resources are nearby. Thus, context denotes additional information that characterizes the situation of an entity (user or item). It is

different to what is traditionally represented in a user model, such as demographics or interests, and refers to "physical contexts (e.g., location, time), environmental contexts (weather, light and sound levels), informational contexts (stock quotes, sports scores), personal contexts (health, mood, schedule, activity), social contexts (group activity, social activity, whom one is in a room with), application contexts (emails, websites visited) and system contexts (network traffic, status of printers)" [65].

Adomavicius and Tuzhilin (2005) describe the traditional recommendation as a function in a two-dimensional space (User×Item) *rec:* $U \times I \rightarrow R$. Lately, Adomavicius et al. (2005) consider that the incorporation of the context can be ensured by exploiting the *Multidimensional RSs* [5]. As a consequence, contextual recommendation can be formalized over a n-dimensional space ($D_1 \times \ldots \times D_n$) as follows (see Figure 13):

$$Rec_{context}: D_1 \times \ldots \times D_n \rightarrow R$$

Where D_i can be, for example Time, Location, Companion

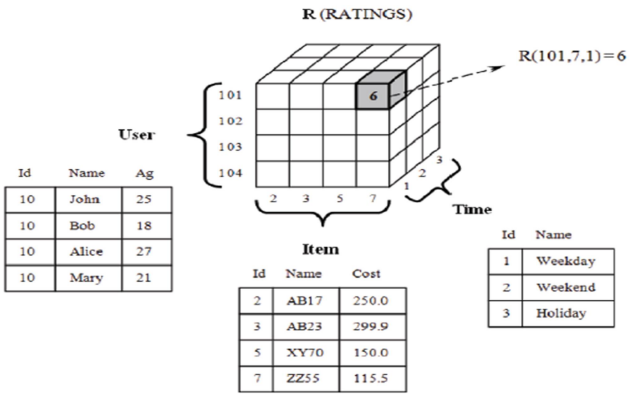


Figure 13: Example of Multidimensional RSs where the temporal context is taken into account.

In fact, the temporal aspect was extensively taken into account in few past works covering contextual recommendation. However, there are other various aspects of the user's situation that can be considered in addition to time; among these aspects we mention the emotional aspect which has received great attention lately in very recent works [66].

The emotional context has definitely a big impact on users' trends and preferences; hence it presents an important dimension that must be taken into account while recommending items to users. For instance, user's mood influences directly his choice about what movie he might want to watch. Furthermore, another dimension that may have an impact on user's choices is his personality. The next sub-section is dedicated to covering techniques used to analyze users' sentiments and personality in order to enhance recommendation relevance.

2.3.4. Personality-based and Sentiment-aware RSs

Since the main function of the RS is helping users to make good decisions about what to choose or consume, it is assumed that user's personality will play a crucial role in the decision making process and it subsequently affects RSs. Furthermore, it has been argued that improving ratings prediction accuracy (by using some metrics such as MAE and RMSE see section 2.5) [67], is insufficient to enhance the quality of a given RS. While, involving the user in the recommendation process by taking into account some user-centric aspects (personality, emotions, opinions, etc.), may have a great improvement on the user's experience.

In fact, the personality is the most prominent factor that determines user's behaviors, trends and interests. By definition in psychology, personality measures individual users' differences in our enduring emotional, interpersonal, experiential, attitudinal and motivational styles [68]. Therefore, these differences are very useful and have to be taken into account by incorporating them during RS design.

Capturing user's personality is similar to build a personalized user's profile on RS field, which is context-independent and domain-independent (i.e. doesn't change in different contexts and different domains such as movies, books, etc.). In [68], John and Srivastava distinguish between five different personality traits: Openness to Experience, Conscientiousness, Extraversion, Agreeableness, and Neuroticism (OCEAN) known as Five Factor Model (FMM). This model is considered one of the most comprehensive and is the most used personality model in RSs [69, 70].

In order to design a personality-based RS two major steps have to be taken into account: 1) personality acquisition and 2) exploiting personality to generate recommendations. For these two steps, different tools and techniques are employed.

Concerning the acquisition of personality techniques they fall into two categories: 1) explicit techniques that employ extensive questionnaires and quizzes depending on the personality model used [71]. Even if these techniques are more accurate, they present some flaws resumed in bothering users and the majority of them don't respond effectively to the asked questions and 2) implicit techniques: which handle issues related to explicit techniques by trying to extract personality parameters seamlessly without disturbing users. To do so, social media streams are used like Facebook [73], twitter [72], and other user-generated data streams (e.g. Email [74], blogs [75]). Even if the implicit methods have advantages over explicit ones, but they require more complex techniques to be able to infer user's personality from his behavior on social media or from the written mails or comments such as *sentiment analysis*, *opinion mining* techniques [76, 77, 78].

Advantages and disadvantages

Introducing the personality within the recommendation process has many advantages over the traditional approaches; its strengths reside in its ability to handle issues related to new-user cold-start [79, 80] and group recommendations [81], since using personality improves the user-similarity computation. Furthermore, it has been shown that users with different personalities tend to prefer novel and serendipitous items [69]; hence using the personality-based recommendation ensures a high level of diversity [82]. Also, this RS approach is more adapted for the cross-domain since it is domain-independent.

Employing the personality in RS context has just been started recently. As a consequence, there are many open issues and challenges that have to be addressed [69]. The step of personality acquisition stills a hard task. On one hand, in order to build an accurate personality profile by using explicit techniques requires an extensive questionnaire which is hard and time-consuming for users. On the other hand, implicit techniques are just at its beginning, and the accuracy of gotten data from social networks is doubtful. Another related issue to personality-based RSs is the privacy. In fact, all the previous research works done in this regard, have not addressed properly the privacy, especially that such recommendation approaches are rested on very sensitive data.

2.4. Challenges and Issues in RSs

As seen in the previous sections, each of the presented recommendation techniques has its flaws or some issues from which it suffers. This section presents the identified issues confronted which must to be handled. By giving the overview of these problems, we can improve the quality of recommendations by inventing new approaches and methods, which can be used as a highway for research and practice in this area.

For each of the discussed challenges, we give a definition, the reasons behind this issue, what are the consequences on the RS, which recommendation type is affected by the issue in question, the challenge to be handled as well as some research opportunities in alleviating these discussed issues and challenges are given at the end.

2.4.1. Data Sparsity

The Sparsity is one of the major problems commonly encountered in almost of all RSs. It refers to the lack of sufficient available data required for recommendation process, in order to infer good recommendations. The main reason behind this issue is the reluctance of users almost of times to give their feedback on items available on the system- even by laziness or they want to keep their privacy- which makes ratings usually scarce [83]. In addition, the large numbers of items in system's catalogue bothers and makes users lazy to give their ratings about [36, 84]. Hence, users' profiles contain typically feedback about a very small proportion of existing items which is insufficient to generate reliable and accurate recommendations [85].

In fact, serving users with good recommendations (i.e. Accurate ones that fit with their tastes), is conditioned by getting to know more about them. This requires acquiring more information about these users (for example feedback as ratings, demographic data such as age, profession, etc…). However, their reluctance to give such information influences badly the processing of the RS and prevents it from computing accurate predictions.

The data sparsity has a bad impact on the overall quality of RS. For instance, the CF approach finds difficulties to make accurate predictions when the rating matrix is very sparse (has missing rating values) [86]. More precisely, pairs of users or items are unlikely to have common ratings, and consequently, similarities computed based only on a small number of ratings, results in biased recommendations [38, 178].

Almost of recommendation approaches are affected by this issue, since they are rested mainly on the user's interactions and are fed by information he provides. On one hand, the CB approach depends on past purchase/like user's history to generate similar content for him as recommendations, hence empty or poor users' profiles don't help to infer accurate recommendations. On the other hand, CF and its variant Demographic (DM) approach suffer from this problem because they are dependent over the rating matrix and demographic information acquired from users in most of the cases [83]. In addition, sparsity can be much more pronounced in context-aware RSs, since they use multidimensional ratings, where it becomes very difficult to provide accurate recommendations for very few rated items [87].

As the main goal of RSs is capturing users' attention and satisfying them, the challenge in that context is computing accurate predictions and serving them with good recommendation even if with just few ratings available. Many possible solutions were proposed to cope with this situation, among them we cite the most commonly employed:

1) The simple way to fill-in the existing gaps in ratings matrix, is to replace the missing values by other default values such as: the middle value of the rating scale (e.g. 3 in 1-to-5 scale) and the average user or item rating [10, 18]. However, this solution shows limitations since it generates biased recommendations.

2) Another more reliable approach is the use of external data and hybridizing two or more approaches under one single recommender. For instance, using content data with the CF approach may compensate the lack of ratings. Hence, the similarities are computed based on content and used in parallel with CF similarities for more accuracy [57, 61, 89]. Another example is using demographic data beside of CF ratings, in order to compute similarities among users based on their own features instead of ratings [5]. The problem with this approach is that these additional data may not be available.
3) One other solution that captured more attention in which many research were conducted, that is the use of dimensionality reduction techniques such as SVD and latent factors [90].

Indeed, in real-world RSs applications this problem is very recurrent, where the sparsity is close to 1 (e.g. It equals to 99.937% in MovieLens Dataset) and it evolves gradually as the number of users and items increases [83]. Many Sparsity measures have been proposed in the literature [91], for instance, in the CF approach, the overall sparsity of the ratings matrix R can be measured as follows:

$$Overall\ Sparsity\ Measure = 1 - \frac{|R|}{m \times n} \qquad \text{(eq.12)}$$

Where:

- $|R|$ is the cardinal of the R matrix which refers to the available ratings.
- m is the number of users and n is the number of items, their product $m \times n$ refers to the number of all possible ratings which are supposed to be provided.

2.4.2 Cold-Start/ Rump-up

Another major serious problem related to the aforementioned issue is the cold-start -also known as the ramp-up problem [20]. It can be seen as a special case of the Sparsity issue [92], where the situation is extremely aggravated by the total absence of any information about either items or users [85]. The cold-start problem denotes the situation when a user or an item has just been added to the RS; where the need to acquire some initial ratings about the newly added object (i.e. User or item) is seriously pronounced, in order to be able to generate reliable recommendations.

Generally, the cold-start problem is threefold [83]: 1) new system: refers to new RS that have been newly launched and it has any information about either its users or items 2) new item-*called also first rater problem*-: occurs when a new item is recently added to RS's catalogue and hasn't received any feedback yet and 3) new user: occurs when a user joined newly the RS and hasn't yet completed his profile either by the ratings or personal information.

In fact, this issue affects both of CF (including DM) and CB approaches. Since community-based approaches (CF and DM) are rested ratings to compute similarities among users or items, it is hard to provide a recommendation in the case of a new user or a new item where any rating is available. However, when it comes to CB approach, it excels compared to CF and DM, because it can provide recommendations on the new-item situation based on items' content and features instead of their ratings as CF approaches do [83]. On the other hand, the major issue of demographic recommenders resides on gathering the requisite demographic information instead of the issue of "new user" that requires the list of ratings of users [20].

Dealing with the so-call cold-start problem is one of the main longest standing challenges in RS field. Having no information about new items or new users, doesn't affect only the recommendations' quality, but it also paralyzes the RS and limits its effectiveness. More precisely, if the required information isn't collected from users about items, no recommendation could be generated. As a consequence, items wouldn't be presented to users, hence the needed information will never be harvested, which a vicious circle [93].

As a consequence, resolving such problem is *very important* for the continuity of the RS. Many researches were conducted in this regard, aiming to deal with the new-item and new-user issues. In fact, both of them may be addressed by using hybrid approaches by using different additional data (such as: content, demographics, etc…) [5]. Especially in new-user situation, other different solutions may be applied in the context of CF approach. Among these solutions, the straightforward one consists in conducting a brief interview with the user in question, during which a set of well-chosen items are shown to him in order to acquire his feedback about them [94].

N.B: since the cold-start problem is a special case of data sparsity problem, all the proposed solution for sparsity still also valid in the context of cold-start.

2.4.3. Long Tail: Popularity Bias

Another issue related to the data sparsity and cold-start (especially new-item issue) consists in the "Long Tail" problem. This term was coined the first time by Anderson in his book [95]. In the book, he categorizes the entire products (items) of a certain market into two main classes: 1) Hits products: representing the best selling products which are very popular, usually sold, and receive a high amount of ratings, this class presents the *head* 2) Niche products which designate less popular products produced and sold for few specialized users, these products present the *tail*. Anderson .C describes the phenomenon that niche products outnumber the hit products largely [96], these hundreds of niches present the so-called long tail (represented by the orange part in Figure 14).

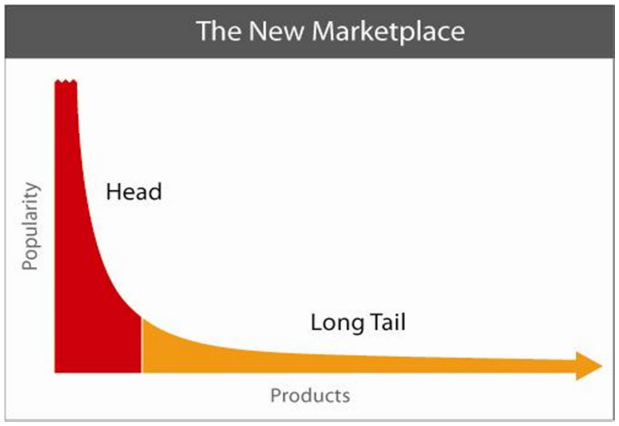


Figure 14: Long tail Problem in markets (Anderson.C, 2006).

In fact, items belonging to the long tail have been ignored for a long time, in the favor of head's items in traditional retailers. However, a strong business must satisfy two main conditions: 1) making all products available to its users and 2) help users to find the wanted products [97]. Hence, the apparition of Internet technologies and online retailers (such as Amazon, Netflix, the iTunes Music Store and so on) afforded access to these niches and facilitates their purchase. [96]. Nevertheless, most of the actual RSs and especially CF ones are rested on the popularity rule and are unable to recommend unpopular items due to the data sparsity problem, i.e. popular items that have received a high number of ratings are prioritized to the detriment of unpopular ones having poor historical data [98]. As a consequence, popular items are often recommended and shown to users, continue to be sold and receive more ratings, in return niche items in long tail still in the shade and are never recommended, which is so disappointing either for users who seek for discovery of new interesting items or for content providers who aim to enhance their sales depending on RSs [20].

Recommending well-known items from a wide range of users is too simple and trivial. The revealed challenge of RSs consists in generating serendipitous and novel recommendations by exploiting the whole of its items' catalog, especially, the long tail part which is neglected almost of times. This will be very

interesting for users who will be supported in their exploration to find out relevant recommendations, as well as for commercial applications which will be benefit from the increase of sales.

Since the long tail issue is a special case of new-item cold-start problem, to handle this issue the aforementioned solutions applied in a cold start situation still valid [85]. Among them we mention the exploitation of items' content, that allows finding relations among items with no or just few ratings based on content similarities, hence the niche items will be very easy.

2.4.4. Gray Sheep

Community-based Recommenders (CF and Demographic techniques) are conceived to classify different users into many classes according to their personal characteristics and/or tastes based on the notion of similarity between them. However, such approaches fail to perform as usual for some users with special and unusual tastes, whose opinions are totally different with other community's users [99]. Hence, they have low correlations with users, don't belong to any of existing cliques of users and subsequently making recommendations for them is nearly impossible [20]. Such users are known under the name of the "Gray Sheep" [100, 101].

The presence of gray sheep users has bad effect on the Collaborative and demographic based recommenders, since it worsen the computation and affect the overall performance of RS regarding the accuracy and the coverage. Fulfilling the needs of gray sheep users is a challenging task, since the aim of such a RS is satisfying its users no matter who they are and how their preferences are. On one hand, handling this issue will increase the loyalty of a RS and retain its users, even those who are very picky. On the other hand, the overall performance will be improved.

Many solutions can be employed in the context of gray sheep; the most trivial is utilizing content data. Since the pure CB approach is rested only on a user's history to infer recommendations similar to what he already consumed, it seems as the best alternative to resolve this issue based on item's content data. In addition, Gray sheep users can be identified and separated from other users by applying offline clustering techniques including k-mean clustering. In this way, performance gets better and recommendation error is minimal [86, 100].

2.4.5. Scalability

Another serious problem related to RS's performance is the scalability, which designates the ability of such a system to anticipate the future growth in the amount of information and its ability to handle it efficiently without any performance degradation [83]. As and when the numbers of items and users increase over time, the computations in traditional CF approach grow in an exponential way and get too expensive which may lead to inaccurate predictions.

With the actual data explosion and the increasing users' demand on online RSs, these last face a great challenge to perform well under these conditions. Robust recommendation approaches have to react immediately to these new requirements. Several solutions were proposed to scale and to speed up computing recommendations. Among them, we mention approximation mechanisms which fail in reaching a good accuracy even if they improve the performance largely [83]. Another effective solution consists in parallelizing treatments over clusters of machines, such techniques are adopted especially by large web companies (e.g. Twitter).

Different techniques have been proposed, including clustering, reducing dimensionality, and Bayesian Network [20]. The problem can be addressed by using clustering CF algorithms that search users in small clusters instead of searching the entire database [9]; by reducing dimensionality through SVD [11], by pre-processing that combines clustering and content-analysis with CF algorithms [24], and by using item classification with weighted slope one scheme in determining vacant ratings in the sparse dataset of CF systems [25].

2.4.6. Overspecialization vs. Serendipity

Recommendations based on the CB approach tend to be "*More of the same*". After analyzing items' features (content) and the active user's history, the RS generates similar items that the user has liked in the past, which might to be less interesting and very obvious [86]. This problem is called the overspecialization, which prevents users to make discovery of other new items that may be unknown for them but still interesting [83].

One of the main reasons behind this problem is the limited content about the items [86], that are represented almost of times by a countable set of features. Especially, when the content is very scarce, as it is the case for many domains such as movies, jokes, books and so forth. In addition, this may lead to represent different items by the same features (keywords) which make distinguishing between them a very difficult task. As a consequence, recommending relevant items is conditioned by the availability of rich sufficient content information to better distinguish items.

Indeed, recommending novel and serendipitous items is very desirable in RSs (i.e. The "help to discover" feature). The serendipity extends the concept of novelty by helping a user find items that he will like, but he has no idea about their existence or he might not have otherwise discovered (handbook 2015) [93]. To overcome the overspecialization in CB approaches, many solutions can be applied. For instance, using genetic algorithms brings diversity to the generated recommendations. As well, we could also benefit from the hybrid approaches by combining CB and CF under a single RS, since CF doesn't suffer from this issue and offer unexpected items to be recommended which is among its strengths [86].

2.4.7. Novelty and diversity

Another two different challenges but somehow related notions are: novelty and diversity [102]. These two aspects are among the required features of a successful RS, which must insure two important functionalities: "help to discover" and "help to explore". On one hand, the novelty of recommended items refers to items that the user hadn't seen previously or he did not know about before and deserve to be discovered. On the other hand, the diversity designates how the members of a set of items are different to each other.

The diversity and novelty are two inseparable and highly desirable functions for modern automatic recommenders. More precisely, if a RS generates a set of items that is sufficiently diversified, then each item in the set is definitely novel compared to the rest. Although, they are rested on two main notions [103]: 1) item popularity: a popular item is an item which is very known and liked by almost everybody. Hence, it is unwise to recommend it will be very trivial and lacks of originality and *novelty*. 2) Item similarity: the low is the similarity between two items, the most they are different to each other, which reminds the notion of diversity.

In fact, CB approaches suffer from an issue of "*portfolio effect*" which refers to the non-diversity problem. An ideal RS mustn't recommend an item that is too similar to what the user had already purchased or consumed in the past [20].

2.4.8. Transparency

The problem with using a RS is whether we could trust the delivered recommendations and to what extent, since RSs are acting such as a black box for their users [104]. The response to this question consists in a major aspect called the transparency. The transparency refers to explaining to users how is the system's functioning, as well as justifying the reasons behind generating recommendations about certain items and not others [105]. Handling this challenge is very crucial for the success of the RS, as it increases the user's trust and satisfaction and help him to make good decisions effectively and in a fast manner.

In fact, CB approaches beat community-based ones (CF and DM) when it consists of transparency. Explaining recommendations generated by CB recommender is an easy task; it can be done by simply

displaying content features on which the RS has made recommendations. However, in the CF approach, for instance it wouldn't be trivial, as the only explanation for an item recommendation is that unknown users with similar tastes liked that item [106].

The challenges of developing effective transparency include (a) choosing whether to trace the workings of an algorithm, or to put together an independent argument that justifies the results of the algorithm; (b) what data to use, including other items, other users, or metadata; and (c) how to present the results to the user [93].

There is not much previous work on evaluating transparency, and often its evaluation is coupled with secrutability [11]. It is however possible to ask if users understand how the personalization works, for example, if they believe recommendations are based on similarity with other items, or on similarity with other users etc.

Users can also be given the task of influencing the system so that it "learns" a preference for a particular type of item, e.g. comedies in a movie recommender system. Task correctness and time to complete such a task would then be relevant quantitative measures [105].

2.4.9. Stability vs. Plasticity

If a RS suffers from empty user's profile which prevents him to infer recommendations, it also suffers from the fact that the user's profile is completely established when the concerned user had rated so many ratings and had given his own preferences. In this last case, the RS is unable to take into account the changes in the user's preferences over time, subsequently it can't change user' profile in question. For instance, an old user who is addicted to the Italian food and became fun of Chinese food, will continue to get pasta and pizza recommendations from CB and even CF or DM recommenders, until he gives some new ratings that may correct the inference. Such problem is known as Stability Vs Plasticity, which is the opposite of the cold-start problem [20].

This problem is faced in RSs that construct a user's profile which gather all long-term users' preferences, such as CB and community-based approaches. However, KB and UB recommenders who interact with the user to learn his need immediately (any changes in his preferences are considered) don't suffer from the stability Vs. plasticity issue since any user's profile is asked to make recommendations.

2.4.10. Time-value: Time-awareness

RSs are typically based on three main entities: users, items as well as the interactions between them (ratings, feedback, etc...). Moreover, the stability of these interactions depends merely on the application domain. For example, in books' recommendations the changes are very slow, since the preferences of users cannot be changed overnight. However, it's quite the opposite daily news or trips' recommendations where the users' preferences are different each time they interact with the RS [93].

The existing recommendation approaches disregard any temporal information during the recommendation process. While these temporal aspect influences the recommendations, since the huge amount about users' interest is gathered over time. Thus, time-aware recommendation approach has to be pushed into the foreground, by exploiting the temporal aspect in order to get more accurate recommendations [107]. Among the main challenges that are faced when we deal with a time-aware recommendation approach is how to incorporate these time information in different situations that may be contradictory and how distinguishing between short-term and long-term users' tastes.

NB: The time-aware recommendations are a special case of context-aware ones, where many aspects other than the time are considered: (mood, palace, etc.) see Section dedicated for context-aware RSs (2.3.3).

2.4.11. Synonymy

A RS is rested only on relationships between users and items; whereas items are treated separately (relationships among them are ignored). As a consequence, classical RSs faced the problem of the synonymy, which refers to recommending the same of very similar items under different forms. For instance, two items described as "drama movie" and "drama film" refers to same item, but are treated by the RS as different ones.

This problem affects CF approaches and also CB ones. The synonymy in CB is related to language's issues (synonymy and polysemy), where two different words mean the same thing. In order to reveal the latent association between items, many techniques can be used, such as: dimensionality reduction techniques, particularly the Latent Semantic Indexing (LSI) method, are capable of dealing with the synonymy problems.

2.4.12. Other challenges

Among the serious challenges faced in recommendation context is the privacy. The most predictive data required about users' preferences are probably the information that users are reluctant to reveal. Because people may not that their personal habit or preferences be known by almost everybody [20].

Another challenge is the necessity to find a certain trade-off between Accuracy and Performance: *Accuracy:* depends on large amounts of data, however large amount of data poses *scalability* issues and increase the *complexity* and subsequently degrades/deteriorates *performance*.

Table 4: Summary of recommendations approaches and their issues.

(✓: the approach isn't affected by the issue and × : the approach is affected by the issue).

Approach	Data Sparsity	Cold-Start		Long Tail	Gray Sheep	Overspecialization	Scalability	Portfolio Effect	Transparency	Synonymy	Stability Vs. Plasticity
		New-user	New-item								
CF	×	×	×	×	×	✓	×	✓	×	×	×
CB	×	×	✓	✓	✓	×	✓	×	✓	×	×
DM	×	✓	×	✓	×	✓	✓	✓	×	×	×
KB	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

2.5. Methods and Metrics to evaluate RSs

In this section we examine how to evaluate the performance of a RS, which consists in its ability to fulfill users' needs. The measures used in the evaluation phase depend merely on the types of data treated and users' requirements [6]. Since the recommendation field derives from Information Retrieval (IR), it is often normal to use information retrieval measures in the context of RSs [108]. Furthermore, some of these measures have to be adjusted to fit the recommendations' field.

Before evaluating a RS we must distinguish between the two different strategies according to which the RS communicates its results to users: 1) best item and 2) Top-N items. The first consists in finding the *best item* that the user will appreciate. In this case, the system tries to compute predictions for all the missing values in ratings matrix and then presenting the item which maximize the predicted rating value. The second consists in finding *a set of items* that the users will probably appreciate; in this case the RS generates an ordered list of the N best items according to their relevance for the user in question.

In fact, in each one of these cases different metrics of evaluation are employed, but generally the evaluation of a RS in the literature is often limited to computing accuracy (i.e. The predictive performance of the RS) [6]. The accuracy generally measures the difference between the values of the ratings predicted by the RS and the real ratings' values provided by the users. Moreover, there are three main strategies conducted in order to evaluate a RS: offline, user studies and online evaluation [109].

2.5.1. Evaluation Strategies

2.5.1.1. Offline evaluation

Generally, RSs have been traditionally evaluated by applying an offline evaluation methodology- Machine Learning Train/Test dataset- that is very simple and safe. This method estimates the prediction error generated by the RS by using an existing dataset which contains information about users and items and transactions between them. The data set in question is divided into two parts using the 5-fold cross-validation: 1) training set on which the prediction algorithm is learnt this part presents the 80% of global data and 2) test set: used to predict the missing ratings value using the algorithm already learnt, this part presents the remaining 20% of the data. This type of evaluation hasn't any risk on disturbing users and makes them quit the system, thus any complex and fluctuating algorithm can be evaluated by this methodology [109].

The offline evaluation doesn't require any interaction with real users; it just simulates their behaviors when interacting with the RS. Some researchers reported that among the limitations of such evaluation methodology the fact that is completely insensitive against changes in users' interactions, so it is unable to measure the impact of such real interactions on the RS. Furthermore, others argue that the quality of a recommender system can never be directly measured because there are too many different objective functions.

2.5.1.2. Users studies: Evaluation by Sampling

This second method consists in nominating a set of some voluntaries users that are asked to perform many tasks requiring interactions with the RS, during the experimentations users are monitored and their feedbacks are saved [109]. After that, participants can be also asked questions about their impressions of the experiments and the RS. This evaluation can answer a wide range of questions. It allows following the behavior of a user during his interaction with the RS, as well as observing if this latter has influenced the behavior of the user. The questionnaire also collects qualitative data to explain the quantitative results.

However, this evaluation method is expensive. It is not always easy to recruit a sufficient number of users, sometimes they need to be motivated by material rewards. In addition, the number of participants is often limited, and subsequently no general conclusions can be drawn concerning a large number of users.

Finally, due to the time constraints of the participants, they cannot be asked to do excessively long tests. Nevertheless, each scenario must be repeated several times to ensure the reliability of the results [109].

2.5.1.3. Online evaluation

Last but not least, the online evaluation implies real RS's users in real time experimentations. It is applied to a sample of users (randomly selected), their reactions are monitored to be compared after that with those of the rest of the users. The risk with using such method is that a user can be lost if the system recommends irrelevant items [109]. For this reason, it is recommended to carry out an evaluation without any risk such as offline evaluation in order to guarantee a minimum quality of recommendation without taking the risk of annoying users or losing them.

The difficulties with the on-line evaluation is to be able to measure the impact of the recommender and only this impact. To neutralize other effect, an *A/B* testing protocol is necessary. An *A/B* testing protocol is a method comparing a baseline control sample *A* used as reference with another test sample *B* which differs on only one parameter. A classic way to analyze a recommender service on a website by *A/B* testing is to design one page containing a placebo recommendation box for a sample A of users, and one page containing the real recommendation box for a sample B of users, and to measure the performance impact between the 2 samples. Unfortunately, *A/B* testing protocol is expensive to design and is seldom used for recommender evaluation. See (Davidson et al., 2010), (Das et al., 2007) for true A/B testing protocol and (Chen et al., 2009) for a similar approach.

2.5.2. Different metrics for offline evaluation

After analyzing the characteristics of each of the evaluation strategies, we found that the first type (offline) is the more appropriate in the context of our work. As we focus in our context on the behavior of a large number of users. The evaluation of user's studies strategy will not be able to conclude with such results. In the current section, we will present the different metrics used for offline evaluation.

3.5.2.1. Rating-based metrics: predictive precision

As in a classic machine learning test protocol, to evaluate the predictive accuracy, the predicted ratings are compared against the real actual ratings on the items. This approach is employed in the "best item" scenario, by using several measures, the most famous being:

- The Mean Absolute Error (MAE):

$$MAE = \frac{\sum_{(u,i,r)\in R} |\widehat{r_{ui}} - r_{ui}|}{|R|} \quad \text{(eq.13)}$$

- The Root Mean Squared Error (RMSE):

$$RMSE = \frac{\sum_{(u,i,r)\in R} |\widehat{r_{ui}} - r_{ui}|}{|R|} \quad \text{(eq.14)}$$

Where:

- |R| is the number of ratings in the test set.
- $\widehat{r_{ui}}$ is the predicted rating value.
- r_{ui} is the real ratings value.

Related metrics are Average RMSE and Average MAE. In this case, we compute the RMSE (or MAE) for each item and then take the average over all items. Likewise, we can compute the RMSE (or MAE) separately for each user and then take the average over all users.

3.5.2.2. Classification-based metrics for relevance

They are known also under the name of decision-based metrics, which serve to evaluate the Top-N recommendations. These measures are taken from the IR field, they don't take into account the predicted ratings values computed by the RS, they consider instead the decision of either recommending a given item or not. The RS is rewarded for the right decisions (when the recommendation list includes relevant items that the user is interested in). On the other hand, the RS is penalized for bad decision (when it integrates irrelevant items in the list of recommendations, or when it lacks of relevant items). The main aim is computing the frequency of good and bad decisions carried out by the RS.

The widely used metrics in this category we find: Precision, Recall and F-measure [6]. We will explain how to compute these measures with the help of the contingency table (see Table 5). Generally, items are classified as being either relevant or irrelevant to users according to their degree of importance; hence we have four different cases:

Table 5: Contingency table showing the classification of the items as relevant or irrelevant.

	Recommended	**Not Recommended**	∑
Relevant	Nb_{RR}	Nb_{RN}	$Nb_{Relevant}$
Irrelevant	Nb_{IR}	Nb_{IN}	$Nb_{Irrelevant}$
∑	$Nb_{Recommended}$	$Nb_{Not\text{-}recommended}$	Nb_{Total}

- **Precision:**

The precision measures the ratio between the number of relevant recommended items and the total number of recommended items, more formally:

$$Precision = \frac{Nb_{RR}}{Nb_{recommended}} \quad \text{(eq.15)}$$

Precision can also be evaluated at a given cutoff rank, considering only the top-n recommendations. This measure is called precision-at-n or Precision@n.

- **Recall:**

The recall measures the coverage of the recommended items, which is the ratio between the number of relevant items recommended and the total number of relevant items, more formally:

$$Recall = \frac{Nb_{RR}}{NB_{Relevant}} \quad \text{(eq.16)}$$

- **F-measure:**

The F-measure is a compromise between the precision and recall:

$$F - measure = \frac{2*Precision*Recall}{Precision+Recall} \quad \text{(eq.17)}$$

3.5.2.3. Ranking accuracy metrics

These measures are useful in cases where the RS have to predict the correct order of the list of recommended items. The relevance of an item is not a binary value (e.g. Interesting or not) but conditioned by its position in the list of recommendations. The majority of the metrics in this category tends to penalize the RS even if the list contains only relevant items, if these items are not well-ordered.

For the most common measures used in this context, we find the two following:

- **The Normalized Distance-based Performance Measure (NDPM):**

This measure is defined as follows [110]:

$$NDPM = \frac{2\,C^{-}+C^{u}}{2C^{i}} \quad \text{(eq.18)}$$

Where:

- C^{-} : the number of contradictory preference relations between the system ranking and the user ranking. A contradiction happens when the system says that the item *i* will be preferred to the item j whereas the user ranking says the opposite.
- C^{u} : the number of compatible preference relations, where the user rates the item *i* higher than the item j, but the system ranks at the same level *i* and *j*
- C^{i} : number of preferred relationships of the user: the number of pairs of rated items *(i, j)* for which the user gives a higher rating for an item than for the other.

The best case is when the RS predicts correctly every preference relation asserted by the reference (i.e. Predicting the position of the item relate to the true user's preference), the NDPM measure gives a perfect score equals to 0. In the worst case, when the RS contradicts every reference preference relation the NDPM gives 1 as being the worst score [109]. Among the advantages of NDPM that it does not penalize the system for system ordering when the user ranks are tied [6].

- **The Normalized Discounted Cumulative Gain (NDCG):**

This measure is taken from IR that is a widely used for evaluation web search and ranked lists, whose aim is measuring the relevance of ranking function that is based on two principal assumptions:

1. Highly relevant items are more useful than marginally relevant items.
2. The lower the ranked position of a relevant item is, the less useful for the user it is, since it is less likely to be examined.

The nDCG is computed based on Discounted Cumulative Gain (DCG) as follows:

$$nDCG = \frac{DCG}{DCG^{*}} \quad \text{(eq.19)}$$

Where DCG^{*} is the ideal DCG computed based on the real preferences of the user, and the DCG computes based on the ranking obtained by the RS is computed as follows:

$$DCG = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{log_2(i+1)} \quad \text{(eq.20)}$$

Where rel_i is the rating of the item at position i in the test set.

In the perfect case, the DCG is equals to DCG* (i.e. the order obtained by the RS is almost the same as that given by the user). The more DCG is nearly equal to DCG* the more the ranking method is accurate. Thus, nDCG falls between 0 and 1, regardless of the test set size.

3.5.2.4. Coverage

The coverage in the context of recommendation systems can be twofold: user-side and item-side. In the first case, the coverage of users computes the percentage of users for which the RS was able to serve by recommendations. In the second case, the items' coverage measures the percentage of items for which a RS was able to provide a prediction among the whole items available in dataset [111]. We are interested only to the second (item coverage), which is the most common and widely used to evaluate RS:

$$Coverage = \frac{n_{pi}}{n_i} \quad \text{(eq.21)}$$

Where n_i is a number of ratings given by the user u_i and n_{pi} is the number of items for which the RS can provide predictions. The more the coverage is bigger the more the RS can be considered of high-performance.

2.5.3. Discussion

The evaluation of the quality of the RS is focused almost of times on its predictive accuracy. This last has some importance but it is not the only thing that matters. Users need to be served by items recommendations that are good enough, but the definition of this "good enough" is domain-dependent. For instance, in news recommender context the user would be interested in the *newest daily* news, however, for a restaurant recommenders the users would be interested by the *nearest* and *cheapest* restaurant and in the other context of movie recommendation the user can be interested to discover the most popular movies, and so forth. Hence, the users' preferences aren't the same in different context and domains.

In fact, beyond the importance of the ability of RS to predict ratings, there are many other crucial criteria that should be taken into account when it comes to measure the effectiveness and the quality of RS [6, 112]. For instance, the **scalability** is a very important point. Furthermore, the ability of the RS to ensure a certain degree of **confidence** by explaining its recommendations (**transparency**) [113, 114] while maintaining users' **privacy** will increase users' **trust** and will have a perceived impact on the quality of the RS. In addition, users are interested to discover surprising items (**serendipity**) that they don't know about before (**novelty**), not similar to what they prefer (**diversity**).

2.6. Review of RSs in Real-Life applications

The use of a recommendation engine is becoming a standard element of a modern web presence. Most large-scale commercial and social websites recommend options, such as products or people to connect with, to users. Every day, we have the opportunity to use one of these systems, but we do not realize that their functioning is based primarily on recommendation algorithms. This section aims to present some of the prominent operational real recommender systems, in order to show their impact in real-life. Our main goal is showing the RS are widespread and can be used in different context and domains. The examples of systems using recommendation process are endless, the list of the reported RSs here is not exhaustive. We present only the most commonly used daily and popular one:

- **GroupLens**

GroupLens is among the first CF systems that were developed and considered as the founder of the CF recommendation approach, it made recommendations about netnews [35, 115]. This system helps their users to find articles that would be likely appreciated in the huge stream of all available items. This RS uses KNN method based on Pearson Coefficient to measure the similarity among the users.

After reading articles, users assign them numeric ratings. GroupLens uses the ratings in two ways [35]: First, it correlates the ratings in order to determine which users' ratings are most similar to each other. Second, it predicts how well users will like new articles, based on ratings from similar users. News reader clients display predicted scores and make it easy for users to rate articles after they read them.

- **MovieLens**

Movielens (www.movielens.org) is a famous, non-commercial, personalized movies RS. It was created by the GroupLens research team for research purposes. The web site offers free access to database logs with different datasets sizes, which are available for research and widely used for benchmarks.

MovieLens owns a large number of movies with very rich data concerning actors, director, etc. It helps also its users to find interesting movies that they may enjoy. To this end, its recommendations are rested on ratings provided by users to build custom taste profile. A variety of recommendation algorithms are employed, the most important is the item-to-item CF algorithm and beside we find the user-to-user CF and the regularized SVD algorithms.

Furthermore, MovieLens employs a preference elicitation strategy to overcome the cold-start problem for new users. To this end, the system asks new users to rate how much they enjoy watching different kinds of existing movies (for instance, they prefer more versus romantic comedies or dark humor).In addition, it allows tuning the matching algorithm so that the results can be more precise to what the user prefer (for example "less violent", "more realistic", or "more ninja").

- **Twitter**

Twitter (www.twitter.com) is the most popular micro-blogging social network in the internet [116, 117]. It permits to its user updating their status and sharing their posts about different subjects and interest (news, entertainment, movies, etc). These statuses are short text messages with limited size (not over 140 characters like SMSs) which are called tweets. The working principle of Twitter is very simple; it is based on the notion of "following" people to be informed about status updates. When a user $\boldsymbol{u}$ follows another user $\boldsymbol{v}$, $\boldsymbol{u}$ is called followers and $\boldsymbol{v}$ is called a follower.

In fact, the great simplicity of Twitter facilitates its wide turnout (over 27 millions of users in 2011). As a consequence, finding interesting tweets and followers became a hard task for users which need to be