Industrial Recommendation Systems in Big Data Context: Efficient Solutions for Cold- Start issues

Ferdaous Hioud

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

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: 9
LIST OF FIGURES: 10
LIST OF ABREVIATIONS 11
GENERAL INTRODUCTION 12
Research Context & motivations 12
Research issues 13
The contribution of the thesis 13
Organization of the manuscript 14
CHAPTER 1:
BACKGROUNDS AND SCOPES 16
1.1. Introduction 17
1.2. Context and Motivations 17
1.2.1. Information access modalities 17
1.2.2. Information Retrieval (IR) 17
1.2.3. Information Filtering (IF) 18
1.2.4. IR vs. IF 19
1.3. Recommender Systems (RSs) 20
1.3.1. Definitions and Terminologies 20
1.3.2. Functionalities and area of use 21
1.3.3. Required data source (Input data) 22
1.3.4. Different Classifications and typologies 23
1.4. General Recommendation process 25
CHAPTER 2:
STATE-OF-THE-ART 27
2.1. Introduction 29
2.2. Review of classical Recommendation approaches 29
2.2.1. Content-based (CB) 29
2.2.2. Collaborative Filtering (CF) 31
2.2.3. Hybrid 41
2.3. Other Recommendation approaches 44
2.3.1. Knowledge-Based (KB) 44
2.3.2. Demographic-based (DM) 45
2.3.3. Context-aware 45
2.3.4. Personality-based and Sentiment-aware RSs 46
2.4. Challenges and Issues in RSs 48
2.4.1. Data Sparsity 48
2.4.2 Cold-Start/ Rump-up 49
2.4.3. Long Tail: Popularity Bias 50
2.4.4. Gray Sheep 51
2.4.5. Scalability 51
2.4.6. Overspecialization vs. Serendipity 52
2.4.7. Novelty and diversity 52
2.4.8. Transparency 52

2.4.9. Stability vs. Plasticity 53
2.4.10. Time-value: Time-awareness 53
2.4.11. Synonymy 54
2.4.12. Other challenges 54
2.5. Methods and Metrics to evaluate RSs 55
2.5.1. Evaluation Strategies 55
2.5.2. Different metrics for offline evaluation 56
2.5.3. Discussion 59
2.6. Review of RSs in Real-Life applications 60
CHAPTER 3:
NEW-USER COLD-START PROBLEM 66
3.1. Introduction 67
3.2. Related works 67
3.3. Our proposed approach 69
3.3.1. Multi-Criteria Recommendation 71
3.3.2. CCSDW: Items' Weighting 71
3.4. Evaluation and validation 73
3.4.1. Datasets 73
3.4.2. Evaluating the goodness of ranking 74
3.4.3. Offline new user profiling experiments 75
3.5. Results and discussion 76
3.5.1. Computing criteria's weights 76
3.5.2. Goodness ranking of different selection methods 77
3.5.3. New user profiling 78
3.6. Conclusion and perspectives 80
CHAPTER 4:
NEW-ITEM COLD-START PROBLEM 81
4.1. Introduction 82
4.2. Our proposed Hybrid approach 83
4.2.1. System's Architecture 83
4.2.2. Content Clustering Module 85
4.2.3. Collaborative Module 87
4.2.4. Hybrid Module 88
4.3. Evaluation and validation 89
4.3.1. Dataset 89
4.3.2. Evaluation procedure 89
4.4. Results and discussion 90
4.5. Conclusion and perspectives 93
CHAPTER 5:
A SCALABLE RECOMMENDATION ENGINE ON LARGE SCALE 94
5.1. Introduction 95
5.2. How RSs meet Big Data challenges? 96
5.2.1. Big Data characteristics 96
5.2.2. RSs vs. Big Data 96
5.3. Related Works 98

5.4. The Proposed Distributed Recommendation Engine 100
5.4.1. About the technology..... 100
5.4.2. The proposed Recommendation Engine architecture 102
5.5. Experimental Results and Performance Tests..... 104
5.5.1. Experimental Environment 104
5.5.2. Evaluation Criteria..... 104
5.5.3. Experimental Results..... 105
5.6. Conclusion 110
GENERAL CONCLUSION & FUTUR PROSPECTS..... 111
Summary of Contributions..... 111
Future Directions 113
***AUTHOR's PUBLICATIONS* 114**
***REFERENCES*..... 115**

LIST OF TABLES:

Table 1: Brief comparison between Information Retrieval (IR) and Information Filtering (IF) systems... 20
Table 2: Comparison of user-based and item-based according to five criteria..... 36
Table 3: Comparison of user-based and item-based at the functional level. 36
Table 4: Summary of recommendations approaches and their issues..... 54
Table 5: Contingency table showing the classification of the items as relevant or irrelevant..... 57
Table 6: Summary of the review on representative real-life RSs. 63
Table 7: The Resulted CCSDw criteria's weights for the YM dataset. 77
Table 8: The Resulted CCSDw criteria's weights for the HRS dataset. 77
Table 9: Comparison between CF and CB approaches. 82
Table 10: Big Data features Vs RS's requirements. 97
Table 11: Performance test for ETL Module under pseudo-distributed settings. 105
Table 12: Performance test for AL Module under pseudo-distributed settings. 105
Table 13: Performance test for Recommendation Module under pseudo-distributed settings. 106
Table 14: Performance test for ETL Module under fully-distributed settings. 108
Table 15: Performance test for AL Module under Fully-distributed settings..... 108
Table 16: Performance test for Recommendation Module under Fully-distributed settings..... 108
Table 17. simulation of real-time user's profiling. 110

LIST OF FIGURES:

Figure 1: Information Retrieval process..... 18
Figure 2: Information Filtering process. 19
Figure 3: Different tasks and goals of a RS..... 22
Figure 4:The two main classifications of RSs in the literature..... 24
Figure 5: Sequential communication between the user and RS..... 25
Figure 6: Main steps to carry out a recommendation process. 25
Figure 7: Abstracted Model for recommendation process. 26
Figure 8: Pure CF approaches: inputs and outputs. 32
Figure 9: Classes of CF approaches..... 32
Figure 10: Hierarchical Clustering algorithm..... 39
Figure 11: Bayesian Network. 40
Figure 12: Hybridization designs according to Burke's Taxonomy (Burke 2002, 2007). 41
Figure 13: Example of Multidimensional RSs where the temporal context is taken into account. 46
Figure 14: Long tail Problem in markets (Anderson.C, 2006). 50
Figure 15: An overview of CCSDW items' selection method and ranking. 70
Figure 16: nDCG@k according to different selection strategies, k= 10, 20, 30,40,60,80 and 100. 78
Figure 17: Presenting how items presented are familiar to users 79
Figure 18: The variation of overall Error (MAE) according to the different selection methods..... 79
Figure 19: Recommendation's Layers. 83
Figure 20: Recommendation process and different components of the proposed system..... 84
Figure 21: Comparison of MAE and coverage between Item-based CF and Content based on CHFSA. .. 91
Figure 22: Variation of MAE and Coverage according to C..... 91
Figure 23: Artificial new item with only few ratings K equals respectively to 2, 5, 10 and 30. 92
Figure 24: Comparison of the proposed method against baseline approaches in new-item settings. 92
Figure 25: Data Flow in Apache Spark..... 101
Figure 26: Apache Spark Stack. 102
Figure 27: High-Level architecture of the proposed Recommendation Engine. 102
Figure 28: Parallelized Recommendation Engine: Workflow Diagram on Apache Spark. 103
Figure 29: The Variation of the Speedup over different modules in pseudo-distributed mode: 106
Figure 30: The Variation of the Parallel Efficiency over different modules in pseudo-distributed mode: 107
Figure 31: The Variation of the Speedup over different modules in Fully-distributed mode: 109
Figure 32: The Variation of the Parallel Efficiency over different modules in Fully-distributed mode:.. 109

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

BACKGROUNDS AND SCOPES **16**
1.1. Introduction **17**
1.2. Context and Motivations **17**
1.2.1. Information access modalities 17
1.2.2. Information Retrieval (IR) 17
1.2.3. Information Filtering (IF) 18
1.2.4. IR vs. IF 19
1.3. Recommender Systems (RSs) **20**
1.3.1. Definitions and Terminologies 20
1.3.2. Functionalities and area of use 21
1.3.3. Required data source (Input data) 22
1.3.4. Different Classifications and typologies 23
1.4. General Recommendation process **25**

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.

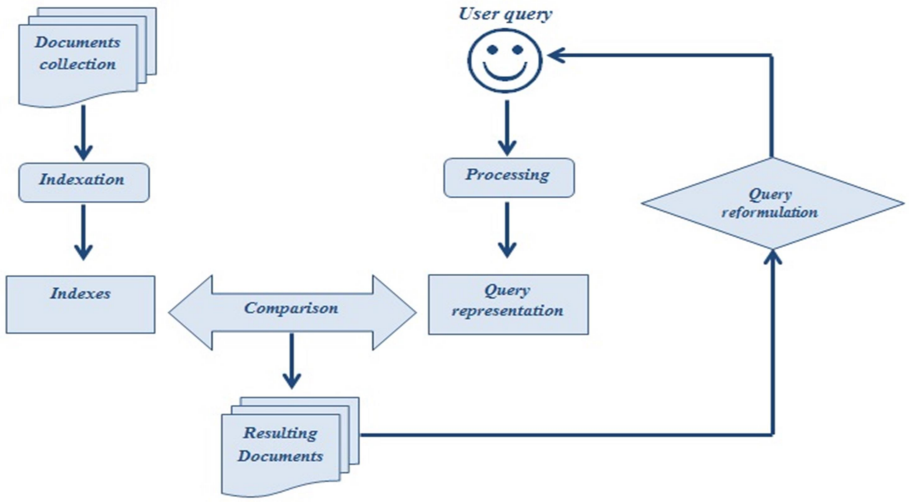


Figure 1: Information Retrieval process.

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.

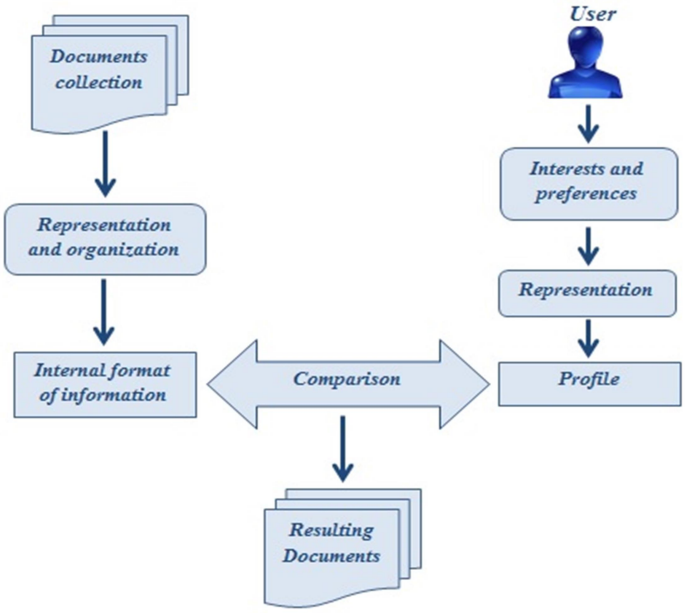


Figure 2: Information Filtering process.

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:

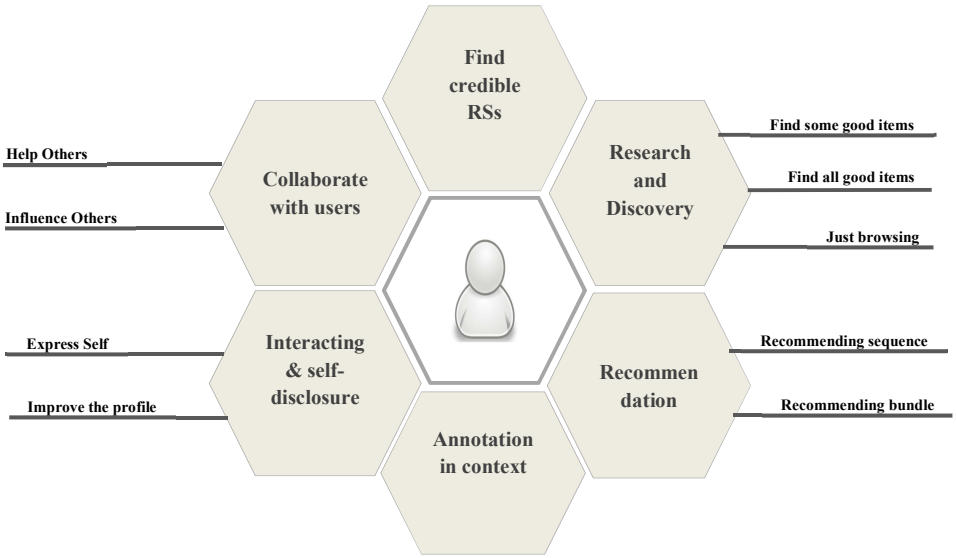


Figure 3: Different tasks and goals of a RS.

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:

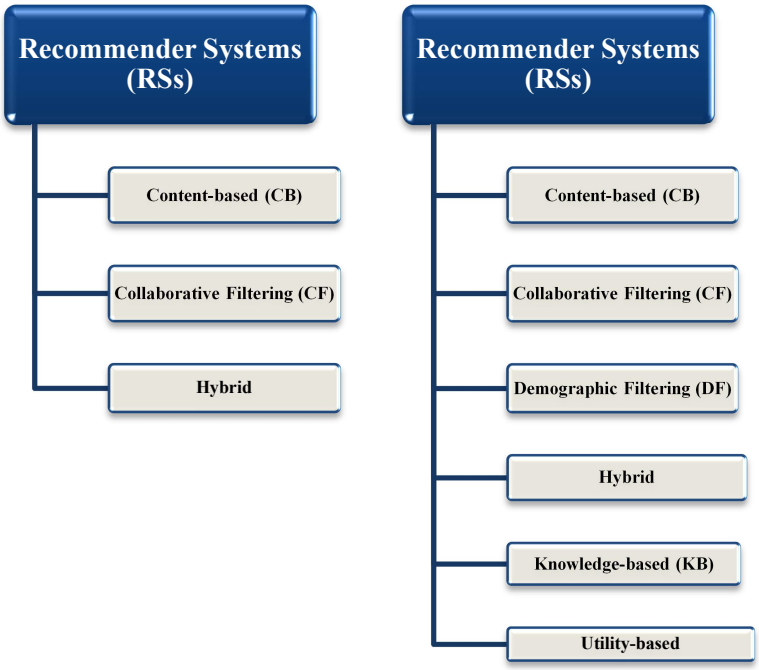


Figure 4: The two main classifications of RSs in the literature.

- ***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.

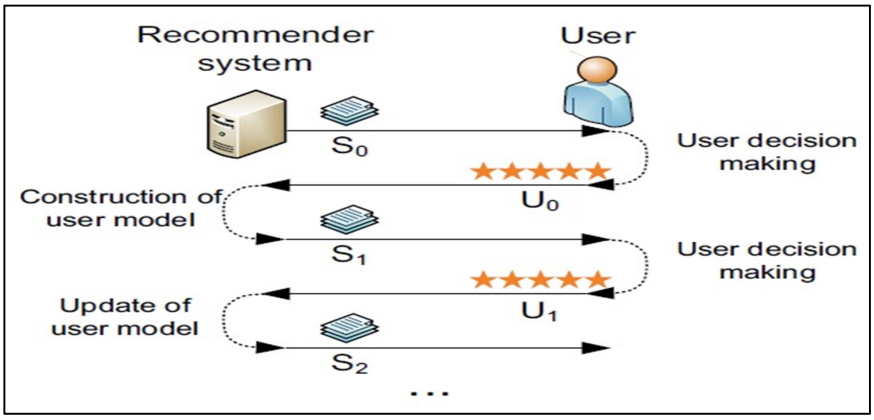


Figure 5: Sequential communication between the user and RS.

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

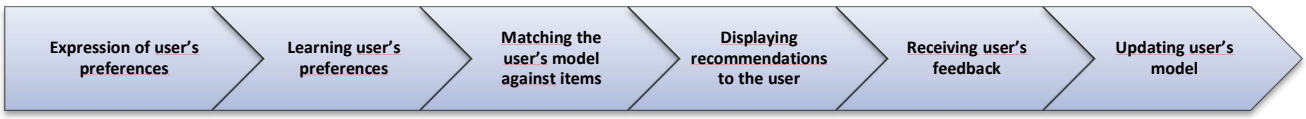


Figure 6: Main steps to carry out a recommendation process.

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:

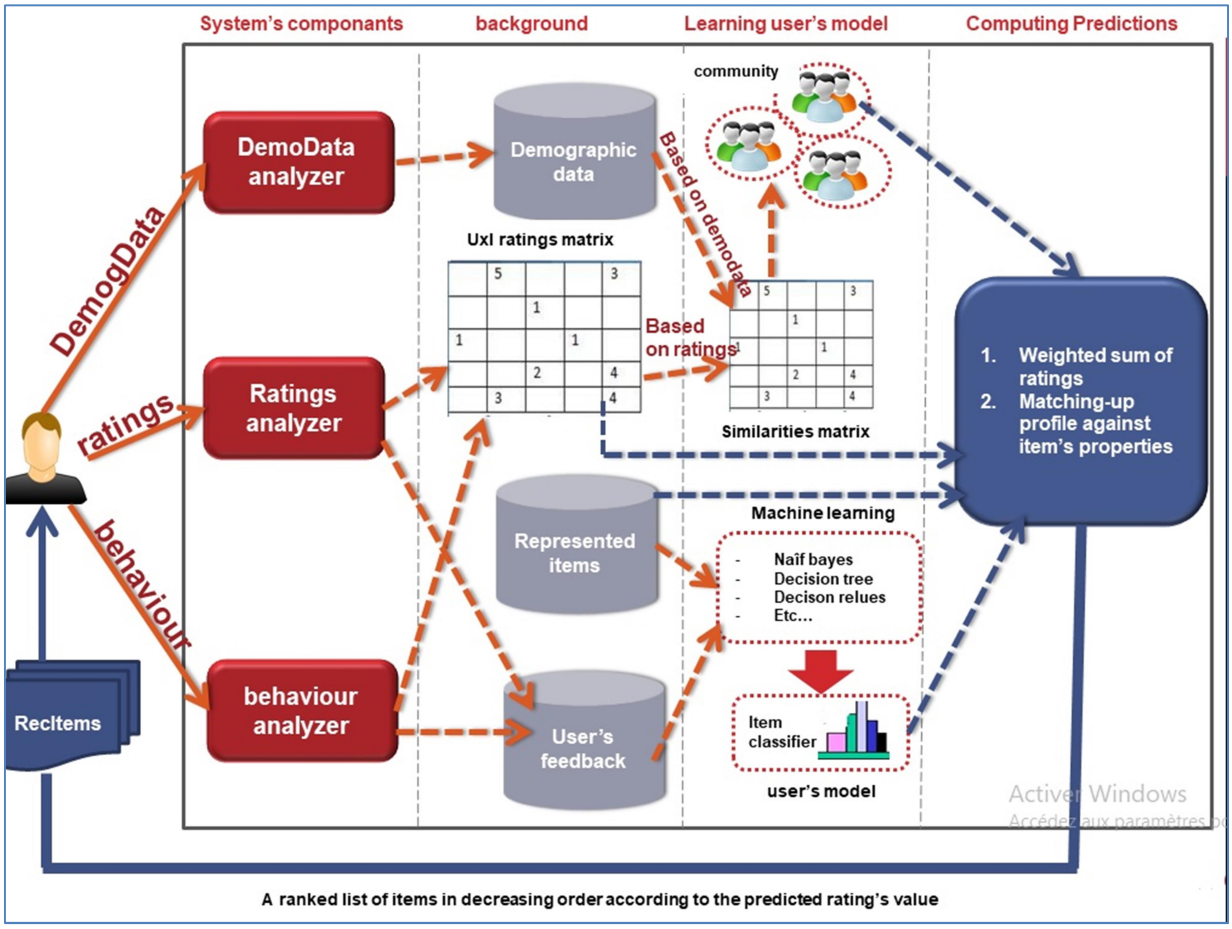


Figure 7: Abstracted Model for recommendation process.

2
STATE-OF-THE-ART

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

Dietmar Jannach, 2011

SUMMARY

STATE-OF-THE-ART 27
2.1. Introduction 29
2.2. Review of classical Recommendation approaches 29
2.2.1. Content-based (CB) 29
2.2.2. Collaborative Filtering (CF) 31
2.2.3. Hybrid 41
2.3. Other Recommendation approaches 44
2.3.1. Knowledge-Based (KB) 44
2.3.2. Demographic-based (DM) 45
2.3.3. Context-aware 45
2.3.4. Personality-based and Sentiment-aware RSs 46
2.4. Challenges and Issues in RSs 48
2.4.1. Data Sparsity 48
2.4.2 Cold-Start/ Rump-up 49
2.4.3. Long Tail: Popularity Bias 50
2.4.4. Gray Sheep 51
2.4.5. Scalability 51
2.4.6. Overspecialization vs. Serendipity 52

2.4.7. Novelty and diversity 52
2.4.8. Transparency 52
2.4.9. Stability vs. Plasticity 53
2.4.10. Time-value: Time-awareness 53
2.4.11. Synonymy 54
2.4.12. Other challenges 54
2.5. Methods and Metrics to evaluate RSs 55
2.5.1. Evaluation Strategies 55
2.5.2. Different metrics for offline evaluation 56
2.5.3. Discussion 59
2.6. Review of RSs in Real-Life applications 60

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):

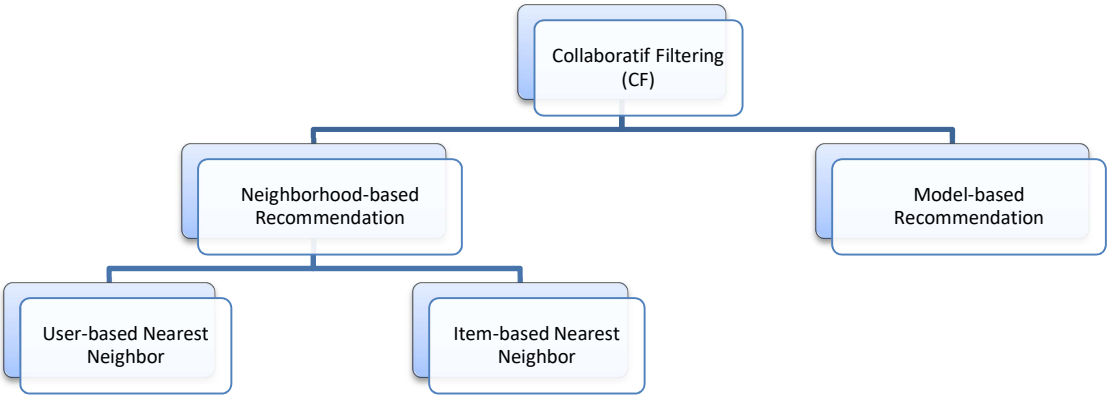


Figure 9: Classes of CF approaches.

- **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:

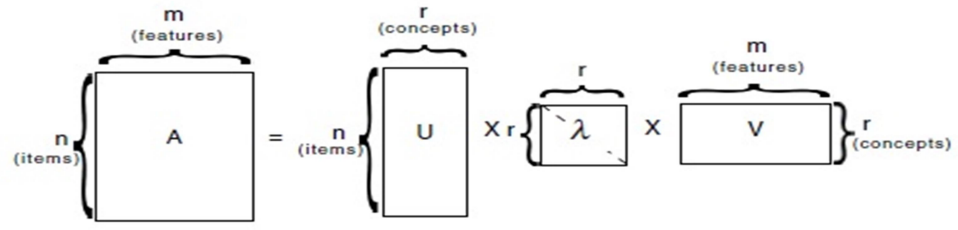
a. Matrix Factorization techniques and Latent Factor models

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].

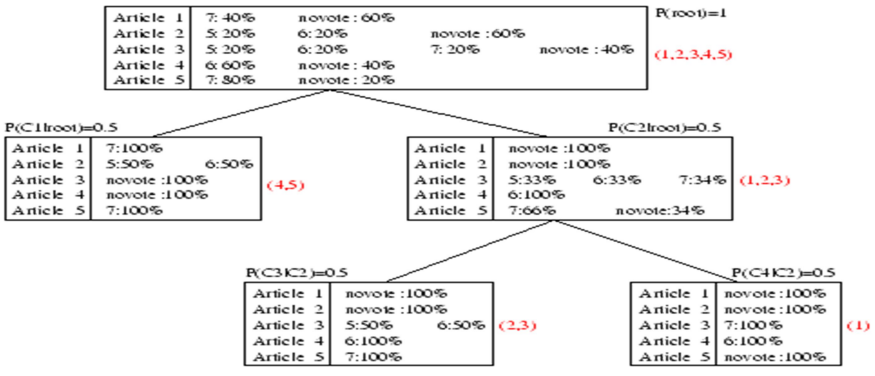


Figure 10: Hierarchical Clustering algorithm.

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).

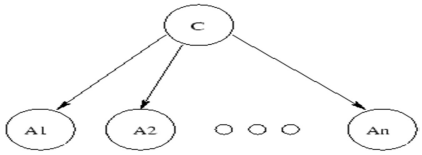


Figure 11: Bayesian Network**.**

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):

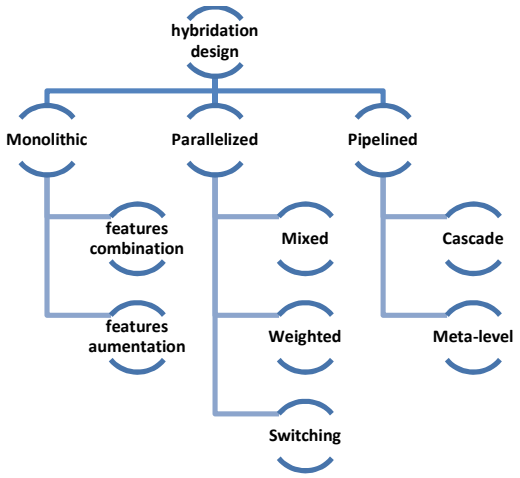


Figure 12: Hybridization designs according to Burke's Taxonomy (Burke 2002, 2007).

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

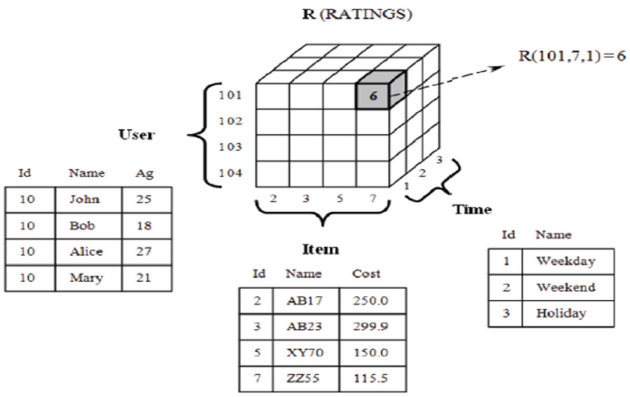


Figure 13: Example of Multidimensional RSs where the temporal context is taken into account.

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).

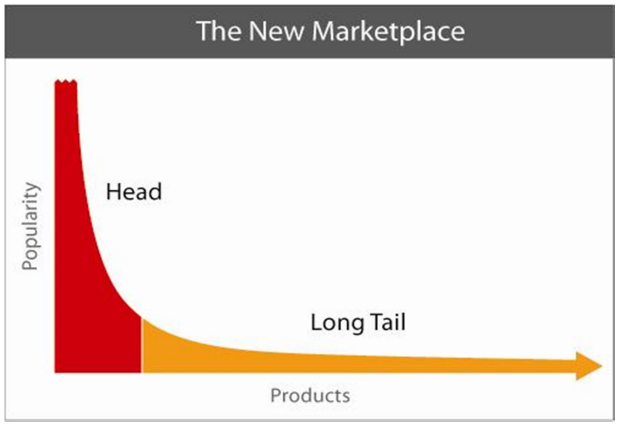


Figure 14: Long tail Problem in markets (Anderson.C, 2006).

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

supported to discover useful content. For these reasons, Twitter employs a RS to build and share user's opinions. This RS is based mainly on content analysis and CF [117].
Furthermore, Twitter is characterized by its extensibility, since it allows API available for developers. For Instance the proposed RS in (Hannon et al., 2010) [116], called Twittomender recommends to users other potential followees, based on their tweets and the social graph in Twitter. To this end, the system uses both content analysis of the tweets and collaborative analysis. Twittomender performs a profiling of each user taking into account his tweets, its followees and its followers, by using seven different profiling strategies that represents users by their [115]:

- Tweets.
- followees' tweets.
- followers' tweets.
- tweets and both their followers and followees.
- followees. IDs.
- followers. IDs.
- followers.ID and their followees. ID.

The first 4 strategies are content-based, the last 3 are collaborative.

- **Facebook**

Facebook (www.facebook.com) is probably among the most known social networks these days (billions of users use it every day). Facebook gives the possibility to its users to create detailed profiles with a rich presentation about themselves, their personal interest and so on. This powerful social network helps to connect people with their friends, other people that they may know and lately it makes them discover more about their interests by using different recommendation algorithms.

To do so, Facebook tracks everything that users when they are connected. It tracks what posts users like or comment on (explicit feedback), what pages they click on (implicit feedback), and many other factors. It also tracks this same information about users' friends and family on Facebook, and then can apply it to users. With the help of this information, Facebook can know what users like and dislike, and therefore suggests things similar to those liked before, and different from what is disliked.

- **Amazon**

Amazon (www.amazon.com) is the most commonly used and popular web site specialized in e-commerce, which offers a panoply of items (books, DVDs, clothes, etc.).

Typically, Amazon is a successful recommendation engine based on Collaborative item-to-item contextual similarities based on purchase history, which was introduced in the early stages of creation of the site [34]. It had popularized many recommendations features such as: "*Frequently bought together*" and "*People who bought these items also bought these items*", which make users discover interesting related items to those searched/ purchased.

Actually, Amazon is considered as a Hybrid RS since its recommendation algorithms are based on some metadata such as genre, author and so on. In addition, it works in push mode with personalized web pages and e-mailing. The main strength point resides in its high performance and scalability over very large users' bases and items catalog. The system is able to react immediately to user's data changes and generate real-time recommendations independently of the user's profile volume [34].

- **YouTube**

YouTube (www.youtube.com) represents one of the famous and most sophisticated industrial RSs. Its aim is providing personalized videos recommendation to its users. Google and YouTube engineers published a paper about how the functioning of their RSs [118]. The paper reports that YouTube RS presents its recommendation as personalized Top-N videos, based on user's activity (watched, favorite and liked videos) and expanding the set of related or similar videos (i.e. Videos watched by the same users), by traversing a co-visitation based graph of videos. A score of "likeness", not necessarily symmetrical, and called "relatedness", between a reference video v_i and another video v_j is given by the formula:

$$r(v_i, v_j) = \frac{c_{i,j}}{f(v_i, v_j)} \qquad \text{(eq.22)}$$

Where $c_{i,j}$ is the count of co-visitation between v_i and v_j and f a normalization function depending on each video. One can use $f(v_i, v_j) = c_i\, c_j$ but other normalization functions are possible.

- **Pinterest**

Pinterest (www.pinterest.com) is an online catalogue used to discover, collect and organize content on the web. Existing content (images, videos, articles, products, etc…) is well annotated thanks to users that give "Pins" to each object visualized, which allows organizing them into boards (i.e. Collections of Pins) by topic. Hence, its name (**P**interest= to **P**in interest). In addition, textual description is provided by used when a Pin is created.

Furthermore, boards reveal relations between Pins: if many users save these two Pins together, there is a high likelihood that another user may find them to be related as well. Generally the recommendation in Pinterest is carried out according to two steps [119]: First, using CF over user curation signals (like co-views and co-likes) related objects are found. Second, the content-based ranking based on visual features, textual descriptions and categories is applied to rank the related objects found in step 1.

- **Netflix**

Netflix (www.netflix.com) is a site equipped with a streaming service which allows its users watching a wide variety of TV shows, movies and documentaries exclusively online. I was founded in 1997 as a mail order DVD rental service, and later it became a powerful online entertainment service. Its main product and revenue source is a subscription service which allows to its subscribers streaming any video at any time [120]. With the unlimited number of available options presented by Netflix, it makes the use of a RS which help users to decide what to watch, where and when more efficiently.
Netflix recommendation algorithms are mainly rested on CF algorithm for prediction of possible rating value that would a user give to an unrated item [4]. They even organized a competition [Netflix Prize 2009], whose aim was improving the accuracy of the rating prediction, which results to a panoply of different algorithms that make up the actual Netflix RS, among these algorithms we find:

- Personalized Video Ranker (PVR): this algorithm ranks videos for each member in a personalized way according to video's genre.
- Top-N video Ranker: the algorithm finds the best few personalized recommendations from the entire catalog.
- Trending now: finds the shorter-term trends from few-minutes to few-days.
- Video-video similarity: an algorithm which falls into the category of "Because You Watched" that presents similar videos to those already watched.

- **LinkedIn**

LinkedIn (www.linkedin.com) is a very common professional social network. The site has claimed that in 2015 the number of users reached 400 million of users from 170 different sectors and among 200 countries. It represents an efficient tool to construct, develop and extend professional relationships, as well as it helps to find jobs, employees, service providers and so on. Particularly, LinkedIn is used by HR firms and employers that are in search of exceptional profiles.

The principle of LinkedIn is based on forming a recommendation concerning different types of items (people you may know, jobs you may like, groups you may want to follow of companies you may be interested in). To this end, it employed extensively the item-to-item CF solution, which is rather than be applied to a specific vertical context, is a horizontal RS that operates several recommendation components. Each of these components is responsible to showcase relationships between items' pairs. For instance, each member's profile on LinkedIn has a "People Who Viewed This Profile Also Viewed" recommendation module. Known as a profile *browsemap*, this module is a navigational aid displaying other profiles that are frequently co-viewed together with the current profile [121]. In fact, collaborative filtering data sets, or *browsemaps*, exist for many entity types on LinkedIn such as member, job, company, and group. These navigational aids are principal components of engagement on the site [121].

To end up this section, we the discussed real-life RSs are presented as a summary table (Table 6), which gives an overview of each RS.

Table 6: Summary of the review on representative real-life RSs.

System	Year	Recommended items	Type			Algorithm	Rating scale
			CB	CF	Hybrid		
GroupLens	1994	News		✓		User-based K-Nearest Neighbor (KNN)	1-to-5 scale
MovieLens	1997	Movies		✓		- User-based K-Nearest Neighbor (KNN) - Item-based CF - SVD-based	1-to-5 scale
Twitter	March 2006	Tweets			✓	Combines 7 strategies for recommendation, 4 among them are CB and 3 are CF	-
Facebook	February 2004	Friends, pages, groups, events, games		✓		An algorithm based on different inputs data: - Personal data - Explicit feedback (like, follow, comment) - Implicit feedback (viewed pages, search history).	-
Amazon	July 1994	books, DVDs, clothes,etc.			✓	Based on item-to-item CF algorithm which uses a different type of metadata	1-to-5 scale
YouTube	February 2005	Videos			✓	Based on item-to-item CF algorithm which uses two input data: - Content data: videos' metadata like title, description, etc… - User activity: explicit attributes, such as ratings and	Binary (like /dislike)

						favorites, and implicit attributes, such as view time.	
Pinterest	Mars 2010	Pictures and photographs			✓	Combines two main algorithms: - Collaborative Filtering - Content-based ranking	Likes (binary scale)
Netflix	August 1997	Movies and TV shows		✓		Many CF algorithms are employed-especially Matrix factorization-, they take into account: - The genres of movies and TV shows available. - User's streaming history and previous ratings he has made. - The combined ratings of all Netflix members who have similar tastes in titles to the active user.	1-to-5 ratings
LinkedIn	2003	Jobs, people profiles, companies and groups		✓		BrowseMaps which is an horizontal item-to-item CF platform constituted of several modules that find co-occurrences between entities in different forms: - Profile browsemap characterized with feature *"People Who Viewed This Profile Also Viewed"* - Job browsemap characterized by the feature "*People Who Viewed This Job Also Viewed*" - Group browsemap - Company browsemap	-

Second Part: Contributions

Towards an Efficient and scalable Industrial Recommendation Engine

3
NEW-USER COLD-START PROBLEM

When a user starts using a RS they expect to see interesting results after a minimal amount of training. Though the system knows little about their preferences, it is essential that training points are selected for rating by the user that will maximize understanding what the new user wants […]

Rashid, Al Mamunur, et al. 2002

SUMMARY

NEW-USER COLD-START PROBLEM 66
3.1. Introduction 67
3.2. Related works 67
3.3. Our proposed approach 69
3.3.1. Multi-Criteria Recommendation 71
3.3.2. CCSDW: Items' Weighting 71
3.4. Evaluation and validation 73
3.4.1. Datasets 73
3.4.2. Evaluating the goodness of ranking 74
3.4.3. Offline new user profiling experiments 75
3.5. Results and discussion 76
3.5.1. Computing criteria's weights 76
3.5.2. Goodness ranking of different selection methods 77
3.5.3. New user profiling 78
3.6. Conclusion and perspectives 80

3.1. Introduction

As we had seen in ***chapter 2***, one of the major issues related to the automatic recommendation is the cold-start which is a complex problem and remains one of the preliminary essential challenges to be handled during the establishment of a RS in industrial context. This problem occurs when the RS fails to acquire enough ratings [10, 38]. The lack of sufficient data about either users or items impacts negatively the quality of the generated recommendations and prevents the system to perform accurately. A study was conducted in [122, 123] quantifying this point by ensuring that the quality of recommendation relies mainly on the quantity of information harvested from users about their preferred items. During our thesis, we raised this issue and we focused on its resolution, by covering its two alternatives: new-user and new-item. The current chapter deals with handling the new-user problem by presenting our proposed methodology, while the next chapter is dedicated to the new item problem.

As we noticed in chapters 1 and 2, all of the recommendation approaches require the presence of large amounts of input data about either users' preferences or items' characteristics. In addition, it is commonly known that RSs work much better for those users on which they have more information about, that are called warm users, i.e. active users who rate items and express their preferences very often. Since a RS is unable to recommend something for a user without any prior knowledge about him. New users present a big challenge for a RS, which has to satisfy them by providing content fitting their preferences. Hence, RSs need to acquire this necessary information to operate correctly and generate accurate recommendations. Generating satisfactory recommendations for new users is conditioned by a crucial step, which is learning profile and ratings elicitation [94, 124]. During this step, it is very important to acquire useful and informative ratings that represent the real preferences of the user to improve the performance of the system.

In general, new-user problem is tackled by applying Active Learning (AL) strategy that resolves the problem from its roots by leading a brief interview with the new user; during which he is asked to give his feedback about a set of *well selected* items. However, the selection of these items is not a trivial task and must to be carried out *carefully*, since the performance of a RS depends mainly on the usefulness of the collected data.

Therefore, our proposed methodology is twofold: First, we propose a novel items' selection approach, based on Multi-Criteria ratings and a method of computing weights of criteria inspired from Multi-Criteria Decision Making (MCDM) approach [125]. Second, the new selection method is exploited to learn new users' profiles, to identify the reasons behind which certain items are deemed to be relevant compared to the rest of the items in the dataset [126].

The section 3.2 presents briefly the different state-of-the-art selection metrics employed to learn new-user's profile to which we compared our proposed methodology. In section 3.3 we present our approach to elicit new user's preferences and learn its profile. The section 3.4 is dedicated for experimentations and evaluation strategy to evaluate the proposed approach, which are conducted on test data sets to show the relevance of our proposition. The obtained results and discussion are covered in section 3.5. Finally, we discuss the contribution of the approach, while giving a number of perspectives in section 3.6.

3.2. Related works

Since, the new-user cold-start problem captures a big attention in the context of RSs era research, many solutions were proposed on its regard. The whole of these solutions can be categorized into two types: 1) User-independent in which the RS handle new users without any intervention from them. Such solutions don't require any interview process which avoids disturbing the user by minimizing his effort. 2) User-dependent in which the new user has to interact with the RS and communicate information either about him or his tastes. Once a RS acquires required data, it can infer personalized recommendations fitting the new user tastes. Our methodology falls into this second category, which is the most efficient and accurate,

as the user is the only one who is able to identify his own preferences, and any explicit feedback from his part is much favored.

The first interview should not be time-consuming: the RS can only ask a very limited number of questions; otherwise the user gets bored and might quit. The process should also be effective, i.e. the answers collected from the user should be useful for constructing at least a rough profile for the user, this ensure that the user won't lose the interest in returning to the RS, due to the low quality of recommendations shown at the first time. In paper [127], the authors observed that the usefulness of collected ratings can be different depending on items rated by each user, hence the necessity to well choose items to be presented during new users' profiling. For these reasons, the process of learning preferences has to be based on two main principles [94]: 1) Minimizing user's effort: by asking few targeted questions 2) Maximizing user's satisfaction: by ensuring accurate recommendations without disturbing him with many questions.

However, RSs deal often with large items catalogs, as well as their task isn't limited only on providing recommendations for new users, but mainly on satisfying them with valuable and accurate recommendations. Hence, the necessity of selecting items to be presented during the new user sign-up, with a *great care.* Nevertheless, the choice of these items is not formalized enough and almost of times still detached of the concerned new user. First of all, the RS have to ask feedback for well-known items that the new user has prior knowledge about; this prevents disturbing the user by asking useless questions that will have no answers. This can be little complicated, since the system have no information about what the new user has experienced in the past. In addition, the system should implement techniques in order to identify valuable items that will improve the recommendation accuracy if they are rated by the new user during the signup step. In the literature, the presented items aren't selected arbitrarily, but under many selection measures based on: information theoretic, statistical aggregation, hybrid or personalized techniques [94, 128,129, 130, 131]. Here we present the most important ones in the literature, to which we will be referred as baseline selection strategies to compare our results. Further techniques and an exhaustive review are surveyed on our paper Hdioud et al., 2017.

- **Popularity** [94, 124]: this measure indicates how frequently users rated an item, as popular items are those familiar ones that receive the most ratings. This category of items captures more attention and receives ratings very often. As a consequence, they are well known by users and they can be rated very easily by new users which minimize their effort [132].The popularity is very easy, inexpensive to compute and we are pretty sure that the user knows already presented items (which minimize his effort on rating them). However, using popularity in eliciting preferences contributes on worsening the existing bias, i.e. popular items garnering even more evaluations. In turn, unpopular items lacking enough user opinions may be hard to recommend. Also, we mention that popular items are less informative about user's preferences in the most cases, because we already know that a popular item is appreciated by a great mass of users, which does not really help to identify real trends of the concerned user.

In fact, items on which we ask the users to rate should not only be familiar to them like popular ones, but also indicative of their tendencies. Thus, it will be very meaningful if we present an item that is subject of conflicting views between supporters and opponents, hence this item will bring useful (discriminative) information. The *contention* is negatively correlated with *popularity*, i.e. popular items are less controversial and vice versa. The most common measures that quantify the contention of an item, which are based on statistics on ratings given by users [122, 123, 133] are as follows**:**

- **Entropy:** represents the dispersion of users' opinions on the item:

$$Entropy\ (item) = -\sum_i p_i \lg(p_i) \qquad \text{(eq.23)}$$

Where p_i : denotes the fraction of item's rating i among different possible ratings' values. For instance, in a 1-5 scale i equals to 1, 2,…, 5. We notice that the major limitation of the entropy is selecting obscure products, e.g. an item which has only two ratings equal respectively to 1 and 5 will have high entropy's value although it hasn't enough of scattered votes. Also, the controversial items are less known by users, and represent only a narrow group of them. Therefore, we shouldn't rely mainly on this strategy.

- **Entropy0** [94]**:** is a variation of the entropy, in which the missing evaluations are treated as a new rating value, which resolves the limitations of entropy. It can be defined as follows:

$$Entropy0\ (item) = -\frac{1}{\sum_i w_i}\sum_{i=0}^{5} p_i\ w_i \lg(p_i) \qquad \text{(eq.24)}$$

For example, in the usual 1-to-5 scale, we add a rating value 0 is added to represent all the missing evaluations, which modify the rating scale. Then, w_i denotes weight of the rating's value i in the 0-to-5 scale. As a result, items with the majority of ratings equals to 0 will have a small entropy0 score and then will not be selected during the sign-up process.

Since the popularity intensifies existing biases and maximizing contention is unwise as it selects obscure items that are rarely rated. Using each strategy separately doesn't bring much information about users' preferences. Therefore, combining both of the popularity and the contention in one measure is favored, as it is the case for *balanced techniques* [94] whose aim is collecting many informative ratings at the same time. There are many hybrid techniques, we consider the following:

- **Log Popularity*Entropy (log Pop*Ent)** [94, 132]: conducted experiences show that the popularity dominates the entropy in the previous metric, since its distribution is exponential when compared to that of the entropy. The logarithm handles the problem caused by the differences between the popularity and the entropy distributions. Another existing variation consists in replacing the log (popularity) by its square root and replacing the entropy by the variance [122].
- **HELF (Harmonic mean of Entropy and Logarithm of Frequency)** [124, 132]: it aims to combine the popularity with the informativeness by using the harmonic mean. To mitigate the differences between entropy and popularity distributions, normalizing them both seems to be successful. The HELF is high when the two operands are also high, which means that neither of them dominates the other.

$$HELF(i) = \frac{2\times LF(i)\times H\prime(i)}{LF(i)+H\prime(i)} \qquad \text{(eq.25)}$$

Where $LF(i) = \log(|Ui|)/\log(|U|)$ is the normalized logarithm of ratings frequency of the item i, and $H'(i)$ is the normalized entropy $H'(i) = H(i)/\lg(5)$.

3.3. Our proposed approach

Our methodology is rested on a selection method inspired from Multi-Attribute Decision Making (MADM); hence we opted for working with multi-criteria RS which is a special case of CF ones. In fact, the multi-criteria problems allow finding the most appropriated items *(*i.e. Items respecting all users' tastes and their selection criteria which are conflicting most of the time), and differences in individual user preferences, which are not explicitly considered. In fact, the most commonly used decision support methods such as outranking methods or the analytical hierarchy process are based on multi-criteria aggregation procedures. Outranking methods determine which alternatives are preferred to others by systematically comparing possible alternatives for each criterion. Subsequently, we have taken inspiration from that to rank the different items based on their attributes and then select the best ones to display for the new user, as they represent the most relevant items.

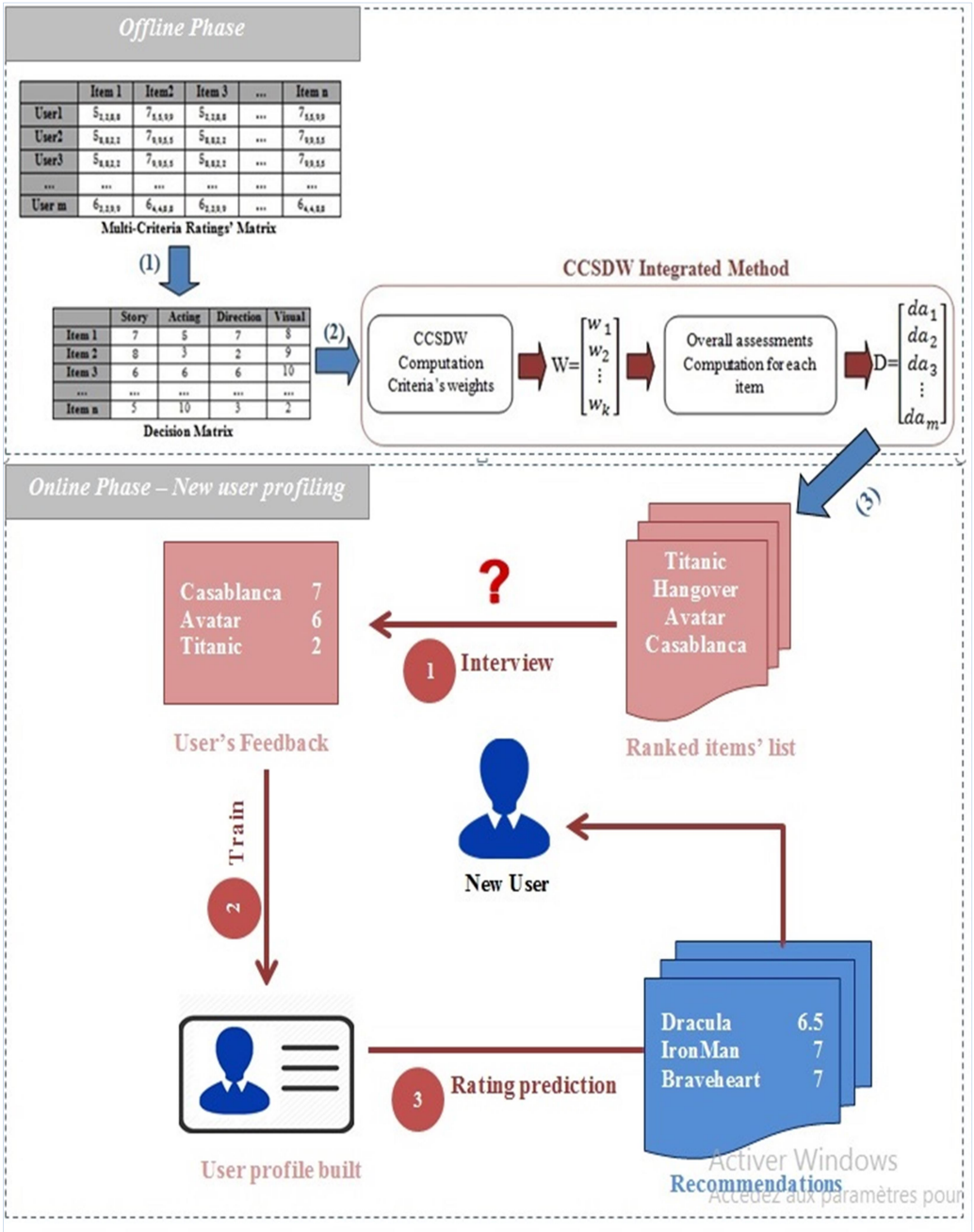


Figure 15: An overview of CCSDW items' selection method and ranking.

An overview of our proposed approach is depicted in Figure 15 which covers two main steps: First, we proposed an objective weight determination method- that we called CCSDW- which provides a methodological items' choice for the RSs. This preliminary step is handled *offline* and consists on computing at first the weights of attributes (criteria) and after that the weights of each item in the dataset. Second, the proposed items' selection method which allows taking into account the dependencies between criteria will be used during the *online* profiling of new users, during which items are ranked in accordance with their weights and displayed to users in question, in order to acquire their ratings on them. The current section will be organized as follows: in the first subsection we present the multi-criteria RSs. Then, the detailed proposed approach is investigated in the two remaining subsections.

3.3.1. Multi-Criteria Recommendation

The goal of Multi-Criteria Recommender Systems (MCRSs) is to find items that maximize each user's utility, just as in the single-criteria RSs. The difference between single-criteria RS and multi-criteria one is that the latter has more information about the users and items, which can be effectively used in the recommendation process. More formally, the general form of a rating function in a multi-criteria recommender system [Mikeli et al. 2013; Sahoo et al. 2006] is:

$$R: Users \times Items \rightarrow R_0 \times R_1 x ... \times R_k$$

Where R_0 *is the set of possible overall rating values and* R_k *represents the possible rating value* for each individual criterion *i (i = 1, …, k).*

The multicriteria collaborative filtering is believed to give better recommendation quality than traditional collaborative filtering because the use of multi-criteria ratings, which can represent user preference better than single-criteria rating [157].

For instance, in Yahoo! Movies[2] recommendation, instead of voting a movie by giving a single global rating, a user is invited to indicate four additional ratings corresponding respectively to Story, Acting, Visual and Direction as they are the four criteria of evaluation. However, in single-criteria RSs, this information would be *"hidden"* within the aggregated overall rating, which may lead to *inaccurate* insights about the true similarity between users' preferences. While in multi-criteria RSs, the information is better identified which permits to detect connections between either users or items.

3.3.2. CCSDW: Items' Weighting

Our method is based on the Correlation Coefficient (CC) and Standard Deviation (SD) integrated approach for determining the Weights of attributes (CCSDW). The CCSDW method determines the weights of attributes by integrating SD of each attribute with their correlation coefficients (CCs) with the overall assessment of items. CCs are determined by removing one attribute at a time from the set of attributes and considering its correlation with the overall assessment of items without the inclusion of this removed attribute. If the CC for the removed attribute turns out to be very high, then the removal of this attribute has little effect on decision-making; otherwise, this removed attribute should be given an important weight.

3.3.2.1. Constructing the Decision Matrix

- ***Problem formalization:*** Let u_1,.., u_n be n users who have evaluated m items i_1, …, i_m based on k criteria C_1, …, C_K. The MCRS problem can be easily expressed in a matrix format as: R which is (n × m × k) matrix of ratings given by each of the n users to each of the m items in k-dimensions. The decision matrix X is an (m × k) matrix in which each element x_{ij} indicates the performance of item i when it is evaluated in terms of decision criterion C_j, (for i =1,2,..., m, and j = 1,2,..., k).
- ***Decision matrix construction:*** As each item is evaluated by different users according to the k attributes (criteria), the R matrix will be transformed to X matrix by computing the performance value x_{ij} as being the average of all rating's values given to the item i according to attribute j by users who voted for.

$$x_{ij} = \frac{\sum_l (r_{li})_j}{L} \qquad \text{(eq.26)}$$

Where:

- $(r_{li})_j$: is the rating given by user l to the item i according to the attribute j.
- L: the total number of users who voted for the item i.

[2] http://movies.yahoo.com/

3.3.2.2. CCSDW integrated Method

This second step as presented in Figure 15 consists on the computation of weights of criteria in the first stage and items in the second. It is mainly based on the decision matrix denoted by $X=(x_{ij})_{m,k}$ which is constructed in the previous step. In our case, items will be the m decision alternatives $i_1, \ldots, i_m$ that will be evaluated in term of k attributes (criteria) $C_1, \ldots, C_k$ and x_{ij} is the performance value of item i with respect to C_j.

Let $W= (w_1,\ldots,w_k)$ be the normalized weights' matrix in such way that $\sum w_j=1$ where w_j is the weight of the criteria C_j. The overall assessment (weight) value of each item (decision alternative) is computed as follows:

$$da_i = \sum_{j=1}^{k} x_{ij} w_j \,.\, i = 1, \ldots m \qquad \text{(eq.27)}$$

The bigger the overall assessment value is, the better the decision alternative is. The best item is the one with the biggest overall assessment value.

By removing criteria C_j from the set of criteria, we define the overall assessment value of each item as:

$$da_{ij} = \sum_{\substack{l=1 \\ l\neq j}}^{k} x_{il} w_l \,.\, i = 1, \ldots m \qquad \text{(eq.28)}$$

The Correlation Coefficient (CC) between the values of C_j and the above overall assessment values can be expressed by:

$$R_j = \frac{\sum_{i=1}^{m}(x_{ij}-\overline{x_j})(da_{ij}-\overline{da_j})}{\sqrt{\sum_{i=1}^{m}(x_{ij}-\overline{x_j})^2 \sum_{i=1}^{m}(da_{ij}-\overline{da_j})^2}} \qquad \text{(eq.29)}$$

Where $\overline{x_j}$ and $\overline{da_j}$ are respectively the mean values of x_{ij} and da_{ij}, for i=1,…,m.

If R_j is high enough and close to one, then the criteria C_j has little effect on decision recommendation. If R_j is very low, then C_j has a significant impact on decision recommendation and items ranking. So, the criteria C_j should be given a very important weight. Based on the above steps, the weight of an attribute is computed as:

$$w_j = \frac{\sigma_j\sqrt{1-R_j}}{\sum_{l=1}^{k}\sigma_l\sqrt{1-R_l}} \quad , j=1,\ldots.k. \qquad \text{(eq.30)}$$

Where the Standard Deviation (SD) is calculated by:

$$\sigma_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_{ij}-\overline{x_j})^2} \,, j=1,\ldots.k. \qquad \text{(eq.31)}$$

The CCSDW algorithm used to compute the weights of both criteria and items is explained in ALORITHM 1.

3.3.2.3. CCSDW: Items' Weighting

After the computation of items' weights, they are sorted in descending order to construct a seed set of items. Thereafter, this items seed set is displayed to the new user; in order to acquire his feedback about the different presented items to learn their profiles. The new user in question is requested to evaluate the items that he experienced or preferred among those that were shown to him. Once this interview process is

achieved; the ratings acquired are exploited to build the new user profile which presents his taste. Then, personalized recommendation could be generated based on the already built profile.

ALGORITHM 1. CCSDW integrated method

Input: X: decision matrix, C: set of k criteria, I: set of n items, W: initial weights of k criteria and D: decision alternatives.
Output: W, D.
1: **for each** of wj in W
2: $w_j \leftarrow 1/k$ // Initialize W in such way that the sum of wj equals to 1
3: **end for**
4: **for each** of Cj in C
5: **for each** of item i in I
6: compute da_{ij}
7: **end for**
8: **end for**
9: **for each** of Cj in C
10: compute the Coefficient of Correlation R_j
11: compute the Standard Deviation σ_j
12: recompute the new weight w_j
13: **end for**
14: **for each** of item i in I
15: compute da_i
16: **end for**

3.4. Evaluation and validation

The evaluation of our proposed approach involves two different experiments:

Experiment N°1: Since the main role of a RS is assisting its users in the discovery of new, various and relevant items. The selected items must respect different users' tastes in order to ensure a low-effort interview process with the user in question. Hence we first evaluate the usefulness of selected items and the goodness of ranking of our proposed method CCSDW.

Experiment N°2: Once the relevance of our proposed method is approved, it will be used in the new-user profiling process. We conduct an interview process, during which we aim to clearly define new user's trends and reasons behind his preferences in order to enrich his profile and then satisfy him with accurate predictions and high-quality recommendations.

To conduct the different tests, a Java program was developed. In which, we implemented all the discussed selection method (including CCDW) as well as the different scenarios for the evaluation. In both of the two experiments, the proposed selection's method CCSDW is evaluated on two large Multi-criteria datasets against other baseline state-of-the-art selection measures presented in section 3.2. In this section, we present the data sets used to conduct our experiments. Then, we present in details the evaluation methodology covering the two different conducted experiments.

3.4.1. Datasets

Our empirical work has been motivated by the availability of two extensive multi-criteria rating datasets from two different domains:

1. The Yahoo! Movies[3] Dataset (YM): Yahoo! Movies is no longer providing multi-criteria ratings; a large dataset was gotten from Nachikita[4] used in his paper [134] that consists of 691511 records. Users could rate movies according to 4 dimensions (Story, Acting, Direction and Visuals) and also assign an overall rating. The ratings' map is given according to a 13-point scale {A+, A, A-,

[3] http://movies.yahoo.com/
[4] http://www.andrew.cmu.edu/user/nsahoo/

B+,…,F} corresponding respectively to {13,12,11,…,0}, while each record of the ratings data has seven attributes: (Item or movie ID (I), User ID (U), Ratings on story (S), Ratings on Acting (A), Ratings on Visual (V), Ratings on Direction (D), The Overall (O) quality of the movie).

2. The Hotel-Review Dataset (HRS): which consists of 878561 reviews from 4333 hotels crawled from TripAdvisor[5]. The user can rate hotels according to eight dimensions: cleanliness of the hotel, the location of the hotel, value for money, quality of rooms, quality of service, quality of check-in, and particular business services, and sleep quality. In addition, they can give a mono-criterion overall rating for a hotel. All ratings are expressed on a 1-to-5 scale. We can observe that the data is very sparse and that only very few ratings per user and item are available. Hence, the data set was cleaned for empirical reasons, and only users with more than 10 ratings are retained.

3.4.2. Evaluating the goodness of ranking

Each of the aforementioned state-of-the-art selection measures (section 3.2) generate a ranked list of items according to the computed measure's value, to be presented to new-users. So, we opted for the use of nDCG whose aim is measuring the relevance of ranking functions. Our goal is to see if selected items go with user's preferences (More precisely, how the different methods of selection could make the right ranking of items relatively to real ratings given by the active user).

In order to evaluate the ranking's relevance of the different selection methods, we proceed as follows:

1) Computing the different selection measures (CCSDW, Popularity, Entropy, Entropy0, LogPop*Ent and HELF) for each item in the dataset, different ranked items' lists are constructed in descending order of the computed measures.
2) Computing DCG which is the Discounted Cumulative Gain, for each ranked list of k items, as follows:

$$DCG = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{log_2(i+1)} \quad \text{(eq.32)}$$

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

3) To evaluate the relevance of each selection method, we compare the ranked list generated by the selection method in question against the reel ranked list according to ratings given by user u_i. For this reason, we compute nDCG@k of a given items' list of size *k* for a given user u_i, as follows:

$$nDCG_i@k = \frac{DCG}{DCG_{ideal}} \quad \text{(eq.33)}$$

Where: *DCG* is computed relatively to the list of k items according to the selection method in question. The ideal DCG_{ideal} is computed in the same way as in equation 27 but for the ranked list ordered by the ratings of a given user u_i.

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

4) Finally, we computed the *nDCG@k* measure for each user according to the different selection measures: CCSD method, popularity, entropy and entropy0. The total value of *nDCG@k* will be the average of all values computed for each user as follows:

$$nDCG@k = \frac{\sum_{i=0}^{|U|} nDCG_i@k}{|U|} \quad \text{(eq.34)}$$

Where |U| is the cardinal of the user's dataset.

[5] https://www.tripadvisor.fr/

3.4.3. Offline new user profiling experiments

This second experiment aims to investigate the suitability of the proposed method in learning new users' profiles. Since the main objective is the satisfaction of the newcomers, which is revolved around providing recommendations with high quality and least effort, we aim to generate recommendations to new users based on the harvested ratings at their first contact with the RS. We conducted offline experiment as a simulation of the sign-up process in which a user is invited to rate items presented to him in order to learn his initial profile. The benefit of the offline experiments is that we can quickly test a variety of strategies without bothering *active users* with items displayed according to different strategies.

To evaluate each of the aforementioned selection methods, we pick randomly 20% of users (as being pseudo new-users) five times, according to cross-validation process, and then we proceed as follows:

1- For each user among the set of pseudo-new-users, we extract his ratings from the training set (which presents 80% of the ratings).
2- We select a ranked list of k items according to the selection method in question, which must be presented to the new users. In order to see how the different strategies perform as long as the system attempted to gather more information about its new-users, we varied the number of movies presented (k=20, 40, etc).
3- For each of the selected users, we reconstruct his new profile as the intersection between user's train ratings obtained in the step 1 and the items presented according to a strategy used obtained in step 2. For each user $u \in U$, the new user profile is presented as:

$$profile\ (u) = \{r_{u,i}\ \in Training\ set\} \cap \{\ r_{u,j/\ j \in PresentedItems}\ \} \qquad \text{(eq.35)}$$

4- Based on the new user profile built in step 3, predictions for non-rated items by the artificial new-users are computed.
5- Once the prediction is computed for the presented items, we evaluate the prediction accuracy with MAE for each user. Then we compute the overall error that is the sum of errors per user:

$$\mathrm{MAE} = \sum_{u \in U} \mathrm{MAE_u} \qquad \text{(eq.36)}$$

Where:

- $\mathrm{MAE_u}$: Error of the artificial new-user u.
- U: The set of all users.

- **Prediction step**

To calculate the similarities between either items or users in MCRS, many multidimensional distance metrics can be used, like Manhattan, Euclidean, and Chebyshev distance metrics (Bilge and Kaleli 2014). Generally, distance and similarity are inversely correlated, in our context, we used the Euclidian distance between items, as follows:

$$sim_{ij}^{c} = \frac{1}{1+ d(i,j)^c} \quad \text{where } d(i,j)^c = \sqrt{\sum_{u=1}^{n} |R_c(i,u) - R_c(j,u)|^2} \qquad \text{(eq.37)}$$

Where sim_{ij}^{c} refers to the individual similarity between items i and j according to the criteria c. $R_c(i,u)$ is the rating given by user u to item i according to criteria c and n is the total number of users.

Then, the overall similarity between items i and j is computed as follows:

$$sim_{ij} = \sum_{c=1}^{k} w_c \,.\, sim_{ij}^{c} \qquad \text{(eq.38)}$$

Where k is the number of criteria and w_c is the weight of the criteria c.

The prediction's step is summarized by predicting the possible rating's value that a user would give to an item. The prediction's value of an item i according to a user u is computed as follows:

$$\text{prediction}_{ui} = \bar{r}_i + \frac{\sum_{j=1}^{l}(r_{uj} - \bar{r}_j) * sim_{ij}}{\sum_{j=1}^{l} |sim_{ij}|} \quad \text{(eq.39)}$$

- l : is the size of the neighborhood of the item i.
- sim_{ij} : is the similarity between the item i and his neighbor j.
- $\bar{r}_i$ and $\bar{r}_j$: refer respectively to the mean' ratings of the item i and item j.
- r_{uj} : is the rating of the item j given by the user u.

- **Recommendation accuracy**

The ultimate aim is to maximize user satisfaction regarding the generated recommendations for him. There are several measures that have been proposed in the literature and used in order to quantify user satisfaction. We used the Mean Absolute Error (MAE) which is computed as follows:

$$MAE = \frac{1}{|T|} \sum_{r_{ui} \in T} |r_{ui} - \widehat{r_{ui}}| \quad \text{(eq.40)}$$

Where:

- $\widehat{r_{ui}}$: is the prediction for item i by user u.
- T: is the set of real value's rating in the test dataset.

3.5. Results and discussion

3.5.1. Computing criteria's weights

The first step of our proposed method consists in computing: 1) the weights of each criterion and 2) the overall assessment (weights) for all items. The results of this step are depicted in tables 7 and 8, for different datasets used.

We can notice that the two criteria *"Acting"* and *"Visual"* are in the lead for the YM dataset (Table 7). This means that the users in our case are more interested by movie's actors mainly which makes a sense since the success of a given movie is related to its stuff. Concerning the second position occupied by "*Visual*" criteria means that in addition to actors, users are interested by visual effects in the movie. The "*direction*" and "*story*" are left to the last stage. These results explain the real trends of users and the reason behind which, several movies have known success despite of the weakness of the story or being directed by a beginner director.

On the other hand, the criterion "Check-in" is prioritized for the HRS dataset. That can be explained by the fact that the clients are almost of the times very tired at their arrival. Hence, they need to validate the check-in process very quickly and effectively. In addition, hotel guests rely on the quality of the sleep in order to rest sufficiently, hence the second position occupied by the "*sleep quality*" criterion. However, the quality of rooms, value and cleanliness don't interest clients very much.

Finally, we compute the weights of each item as an overall assessment based on criteria' weights already computed. Obviously, the bigger the weight is, the more important the movie is. Thus, we rank all items on a descending order from the most significant to the less ones, according to the computed value under ranking function: CCSDW, popularity, entropy and entropy0, LogPop*Ent and HELF. The results are presented and discussed in the next sub-section.

Table 7: The Resulted CCSDw criteria's weights for the YM dataset.

Criteria	CCSDw weights
Acting	**0.272508**
Visual	0.268124
Story	0.237589
Direction	0.221779

Table 8: The Resulted CCSDw criteria's weights for the HRS dataset.

Criteria	CCSDw weights
Check-in	**0.188311**
Sleep Quality	0.166752
Business Service	0.162152
Location	0.13065
Service	0.0959185
Rooms	0.0885002
Value	0.0862971
Cleanliness	0.0814184

3.5.2. Goodness ranking of different selection methods

Figures 16.a and 16.b present the evaluation of each implemented selection method by nDCG@k metric for (HRS) and (YM) datasets, where k takes respectively values: 10, 20, 30,40,60,80 and 100. Since the user is hurry to find what he likes, it is rare that he browses the entire items' list. Once the user is annoyed, he leaves the system as soon as possible. We assume that the most relevant and important items must appear on the top of the ranked list displayed to new users. Therefore, we evaluated the ranking functions only for the Top-k items.

As we mentioned already the more nDCG value is bigger (tends to 1), the more the ranked list is accurate. The results show that the *CCSDW* method outperforms perfectly all the other methods. Furthermore, we notice the following:

- The relevance is enhanced (nDCG approaches nearly 1) with the increased value of k. This reinforces the idea that the more we get more information about the more accurate users' profiles are constructed.
- The HELF shows the same behavior to CCSDW on the early stages (k equals to 10 and 20) but our proposed method excels beyond k=20.
- The *Entropy* doesn't perform well compared to the others selection methods, as it shows minimal NDCG values, as it selects special items characterizing their unique tastes. This meets well with what is known about Entropy that is not suitable to be used alone during sign-process to acquire users' ratings.
- The *Entropy0* and *Popularity* show almost the same behavior, but *Entropy0* stays better than *Popularity*, which is absolutely normal since it is considered as a balanced technique that combines the *popularity* and the *contention* in the same measure. We conclude that users prefer to discover divers kinds of items (popular, controversial and hybrid ones) and not only one category of them.

To summarize, our proposed method CCSDW outperforms other methods in term of the nDCG value for any value of k and for the two used datasets. These results confirm the effectiveness of our method on ranking items which approaches users' tastes.

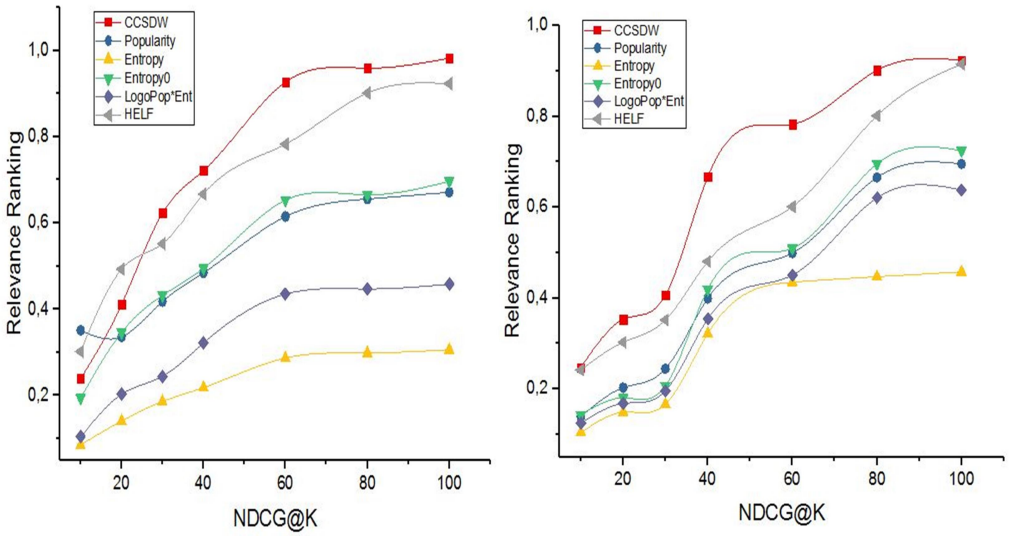


Figure 16: nDCG@k according to different selection strategies, k= 10, 20, 30,40,60,80 and 100.

Fig 16.a for (HRS) Dataset and Fig 16.b for (YM) Dataset

3.5.3. New user profiling

In our approach, we focused on two major aspects to bootstrap new-users: 1) minimizing user's effort during the sign-up process, which concerns mainly if the user recognizes easily the items presented to him, to be able to rate them, and 2) maximizing the user's satisfaction in term of quality and accuracy of generated recommendations.

- **User effort:**

Figures 17.a and 17.b show the number of recognized and rated items by users relatively to the total number of the shown items according to different selection's strategies (i.e. The number of ratings per each constructed new-user profile). The higher is the number of recognized items, the less is the effort required from the user during the sign-process. So, we may notice the following:

- The CCSDW method performs nearly in a similar way to the popularity method in first stages, but it outperforms lately. It is well known that the popularity is perfect to minimize the user's effort as it selects popular items easy to be recognized.
- The entropy method doesn't give satisfaction, as the first selects obscure items (presenting a narrow users' group), subsequently it evolves the user to make more effort in order to give his feedback about shown items.
- The balanced methods such as Entropy0, LogPop*Ent and HELF takes respectively 3rd, 4th and 5th positions after CCSDW and Popularity. Hence, they require a medium effort from the user.
- In summary, the proposed method CCSDw minimizes perfectly the new user effort compared to baseline selection methods. However, many researches [Rashid et al, 2008; Cremonesi et al, 2012] had shown that users pay more attention to the quality of the offered recommendations and they ignore the effort if they are satisfied by the RS. The average number of ratings collected from a user is almost of times 20 or 30 [Rashid et al, 2008], which is acceptable by the user. Therefore, good elicitation methods have to make a compromise between good recommendation and an acceptable effort.

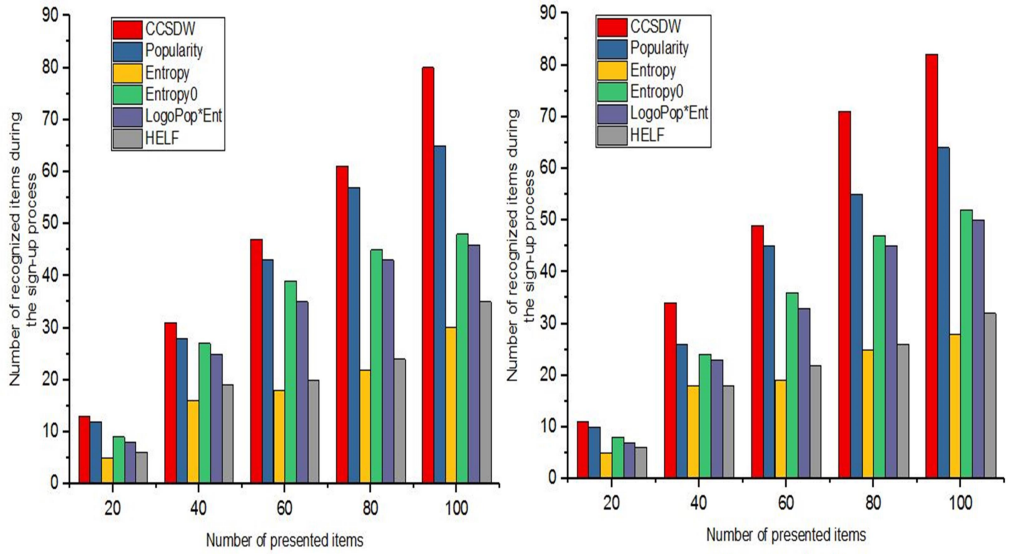


Figure 17: Presenting how items presented are familiar to users
Fig 19.a for for (HRS) Dataset and Fig 19.b for (YM) Dataset.

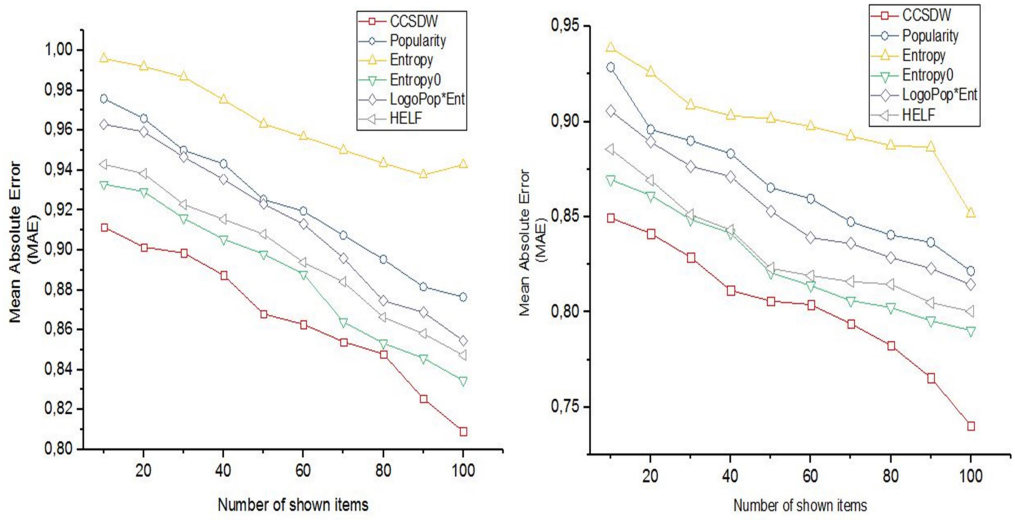


Figure 18: The variation of overall Error (MAE) according to the different selection methods
Fig 18.a for (HRS) Dataset and Fig 18.b for (YM) Dataset.

- **Recommendations accuracy:**

Figures 18.a and 18.b present the variation of the overall error's measure, according to different selection's methods used to learn new-users profiles. The x-axis presents the number of movies shown to the user, and the y-axis presents the MAE's value. The first observation made, is that the more we show movies to the new user the more the error value decrease. This proves that the more RS knows about its users the more it makes accurate recommendations for them.

On the other side, we observe that the *entropy* makes weak recommendations on term of accuracy compared to others. These results are perfectly normal, as we cannot rely primarily on these methods for the following reasons:

- The major limitation of the entropy resides in the fact that it selects obscure products, e.g. an item which has only two ratings corresponding respectively to 1 and 5 will have high entropy's value although it has not enough of scattered votes.
- The controversial items are less known by users, and represent only a narrow group of them.

Regarding the *popularity*, we notice that it ensures slightly accurate recommendations compared to entropy. This is because popular items are familiar ones, so it is more likely that the new user recognize and rate them. Consequently, the more information we have, the more we can generate specific and accurate recommendations (hence the low value of MAE).

On the other hand, popular items don't bring big information about the user's preferences in the most cases. Because we already know that the item is appreciated by a great mass of users, which does not really help to identify real trends of the user in question. Also, we notice that using popularity to elicit preferences worsens the bias that is popular items garner even more evaluations.

3.6.Conclusion and perspectives

Since Recommender Systems work much better for those users on which they have more information about, it is of great interest to get users' hidden preferences. The majority of solutions that had been proposed in the literature focus mainly on conducting an interview process with new users, by asking their feedback against a set of some well selected items. However, the interview remains non-formalized enough, since the reasons behind the items' selection remain unknown.

Active Learning (AL) and Rating elicitation strategies have been addressed extensively in many previous works, but they focused especially on classical CF recommenders. However, in our context, we extended the concept of mono-criterion ratings to multi-criteria ones, to meet the requirements of more practical RSs. We assume that they contain much hidden information about the real preferences of users. Hence, identifying which criteria is more important for one category of person may help to well choose the items to be presented for a new user during the sign-up stage.

Our approach based on CCSDW method is evaluated in term of relevance of selected items and its usefulness during the interview process with the new user. It was compared to some other baseline methods used in the context of learning new users' profiles, to illustrate its potential application. The evaluation of different strategies according to the number of ratings acquired, takes into account two major aspects: the ratings prediction accuracy measured with Mean Absolute Error (MAE) and the goodness of the recommendations' ranking, measured with Normalized Discounted Cumulative Gain (NDCG). The CCSDW has shown a significant improvements in accuracy when the bootstrap process was driven. The obtained results evidenced that our approach has shown a variety in the selected items' set that fits with different tastes.

Future work aims to ask a user well targeted questions concerning criteria interesting him instead of asking him about the items. In this way, we can retrieve more information while asking only a few questions. In fact, a recent work reported in [132] group ratings elicitation strategies as being: personalized or non-personalized. The first consists of selecting the same items' set to be presented to each new-user during the sign-up process. Concerning the second, the items set is tailored according to the user in question. This presents another novelty of our proposed method. In fact, basing the items' selection on the criteria's weights allows good control of selection by the user and subsequently leverage the quality of selected items. At sign-up stage, instead of asking users their feedback on a set of non personalized items, we ask them about the most important criteria to them and to what extent. Then a set of personalized items set can be generated and presented to acquire their ratings for them. Hence, our selection method may operate according to two modes: personalized and non-personalized.

4
NEW-ITEM COLD-START PROBLEM

A particular problem for cold start is when there is an item with little to no ratings as CF will be unable to recommend other items based on the item with few ratings […]

The new item problem occurs when a new item is introduced in the database because the item has no ratings. The item will therefore be unavailable as a recommendation for the users until it has gathered some ratings. […]

Johansson & Runnman 2017

SUMMARY

NEW-ITEM COLD-START PROBLEM 81
5.1. Introduction 82
5.2. Our proposed Hybrid approach 83
5.2.1. System's Architecture 83
5.2.2. Content Clustering Module 85
5.2.3. Collaborative Module 87
5.2.4. Hybrid Module 88
5.3. Evaluation and validation 89
5.3.1. Dataset 89
5.3.2. Evaluation procedure 89
5.4. Results and discussion 90
5.5. Conclusion and perspectives 93

4.1.Introduction

This chapter is focused on handling the new item problem and its underlying issue, known as long tail referring to items with low demand [see section 2.4.3]. As seen previously, CF approach is the most successful approach in recommendation field and it is also much favored over CB approach. However, it suffers from many serious problems related to data sparsity that affect recommendations' accuracy. Specifically, when a new item is added to the items' set, the system fails to perform well due to the insufficiency of available information about this item (i.e. Ratings). More precisely, computing predictions to this new item to be recommended latter seems impossible, and thereafter users will never be aware about its existence and would never rate it. To get out of this vicious cycle, this issue has to be handled seriously, since users use RSs to discover new products continuously.

Various solutions to tackle this so-called new-item cold-start problem have been proposed in the literature, but the most common thread in RSs research field is the hybridization. This latter consists in combining two or several recommendation techniques under one single RS –called Hybrid RS- to achieve a high-level performance. In our context, we aim to benefit from the *content* to bridge the gap between the existing items and those newly added by building relationships among them. By doing so, the lack of sufficient ratings will be offset by computing similarities between items based on their whole raw content instead of their ratings.

As the computation of items similarities is independent of the used method for prediction generation, multiple knowledge sources about items can be used to determine similarities between them [136].Our hybridization proposal was motivated by the existing complementarity between CF and CB approaches as presented in Table 9. On one hand, in CF techniques the prediction computation is based primarily on the concept of similarity between either users or items; also the quality of recommendation is mainly based only on the amount of users' ratings gathered and any additional information such as *content* is completely denied. On the other hand, the highlight of CB RSs is that *new items* can be immediately recommended once their attributes are available, then the aforementioned problem is not placed. However, these techniques inherit the classical problems of natural language ambiguity, since they usually use textual features in their items' representations and don't take into account any semantic knowledge about items. For this reason they tend to achieve a low-accuracy compared to CF ones and they are seldom used alone in practical cases [5]. Hence, classical CB approaches need some improvements to consider semantic aspects while representing content, which could be helpful to better understand items' relationships and would enhance recommendations quality.

Table 9: Comparison between CF and CB approaches.

Collaborative Filtering (CF)	Classical Content-Based (CB)
New-item problem	No New-item problem
Serendipity & novelty	Overspecialization
Based on ratings and user's interactions	Based on content available for each item
High accuracy	Low accuracy

In fact, employing classical CB techniques in combination with CF ones, to tackle sparsity, cold-start and long tail problems is a popular solution. There were many studies conducted in this regard [138, 139, 140, 141, 142], but the notion of the content differs from one to another. Generally, the content can be any additional information about either an item and/or a user, such as item's features, demographic data, etc. However, up to now any previous work has operated on a broad and consistent content of items. Furthermore, the unstructured keyword-based representation adopted almost of times in classical CB approaches has two main flaws [136]:

- It results in a substantial amount of *noise.*

- It is unable to capture the complex relationships among items at a deeper *semantic level*, based on the inherent properties associated with these objects.

These gaps lead to inaccurate recommendations, as a consequence to be able to recommend different types of complex items using their underlying properties and attributes, the RS must be able to rely on the characterization of objects, not just based on keywords, but on a deeper level [137].

Therefore, the aim of our proposed method is twofold [143]: First, we introduce a powerful items' content clustering algorithm; which uses a Hybrid Features Selection Method (HFSM). This method combines statistical and semantic relevant features to get the maximum profit from items' content. We assume that this content clustering will overcome the problems related to classical CB approaches, so it can ensure a high accuracy and can be used alone. Second, the proposed content clustering is linearly combined with item-based CF approach under a single hybrid RS, in order to show its ability to replace-or to alternate with CF approach, when this last is unable to perform as expected (i.e. When it lacks sufficient data), as well as, to tackle new-item and long-tail problems.

The section 4.2 presents the architecture of our proposed hybrid RS in details, covering its different components. The section 4.3 is dedicated for experimentations and evaluation strategies to assess the proposed hybrid approach. The discussion of the obtained results is covered in section 4.4. Finally, we discuss the contribution of the approach, while giving a number of perspectives in section 4.5.

4.2.Our proposed Hybrid approach

4.2.1. System's Architecture

Our proposed system consists of four main modules (components); the following sub-sections describe it in details. First, we present the architecture of the system in layers; with different tasks insured by each one (see Figure 19). Second, the whole recommendation process is discussed deeply (see Figure 20).

5.2.1.1. Recommendation layers

The recommendation layers of the proposed system consist on three ones (Figure 19):

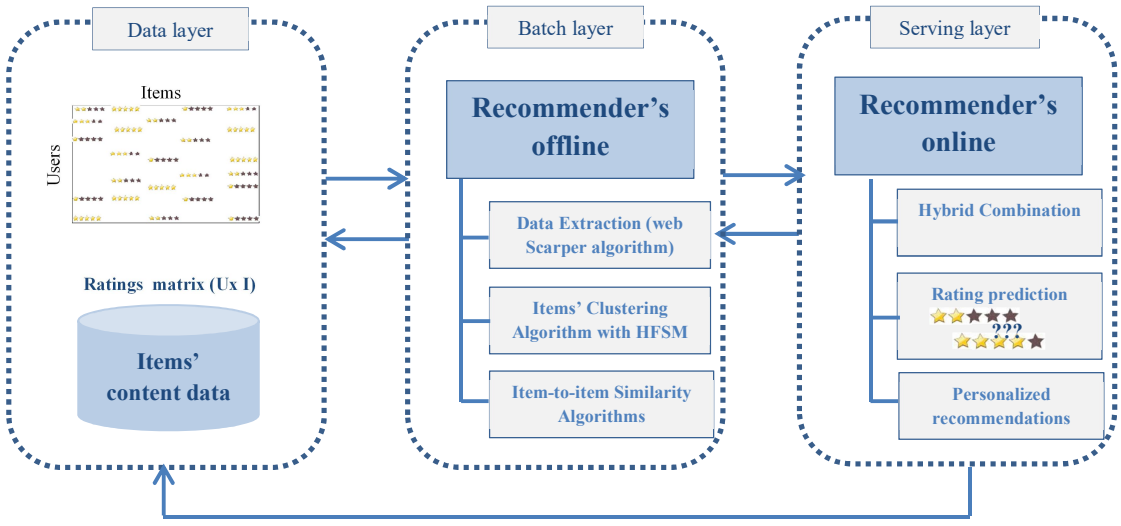


Figure 19: Recommendation's Layers.

- **Data layer:** is the first primary layer of the system. It is the module that stores two types of information: 1) ratings matrix (expressed in Users X Items space) and 2) a consistent items' content stored in files (one file per item), all of these data are exploited by the second layer.

- **Batch layer:** components of this layer communicate with the first one, to get the necessary information needed during the performed treatments. This layer executes three heavy offline tasks:
    1. Content data extraction: an extensive content about each item is extracted and stored in files separately (one file per each item).
    2. Items' clustering : the items' content retrieved previously and stored in the data layer, is exploited during the proposed clustering of items (the algorithm is covered in the next section)
    3. Similarity computation: item-to-item similarities are computed using two distinct methods: item-based CF and the proposed content clustering.
- **Serving layer:** the entire tasks of this layer are executed online. The role of this layer is serving users by providing them personalized recommendations. It relies mainly on hybrid item-to-item similarities computed based on those generated previously by the batch layer.

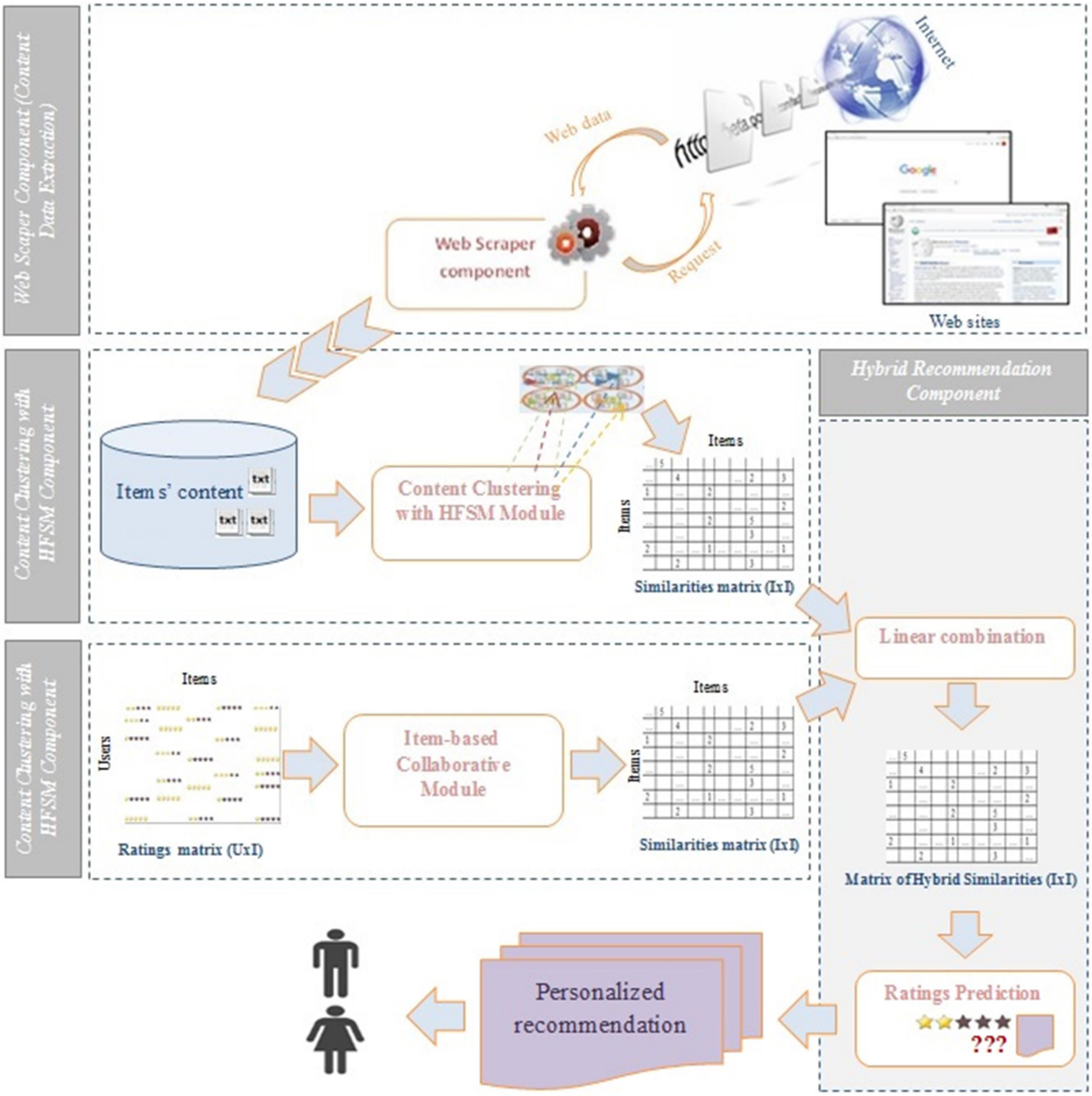


Figure 20: Recommendation process and different components of the proposed system.

5.2.1.2. Recommendation process and System components

First, we developed a Java web scarper program to retrieve an extensive content about items automatically. In our case, we used a data about movies, on which we have only little information (some features like title and category). For each item (movie) the web scarper module has to emit a search query via Google[6] (the title of the movie in question). Among the obtained results, the web scarper keeps only those corresponding to Wikipedia[7] and Rotten Tomatoes[8]. Then, it extracts the synopsis of the movie and stores it into a file.

The second module is content module which ensures the clustering of items with Hybrid Features Selection Method (HFSM). The third is the pure item-based CF module that depends only on users' ratings to compute the similarities between items. Both of the two first modules generate an item-to-item matrix similarity. Finally, the fourth module combines the two similarity matrices generated previously into another one. This permits to compensate the missing values in the CF similarity matrix with the others based on content clustering. The resulting matrix will be used for the prediction and the recommendation steps. The main operational aspects are depicted in Figure 20.

4.2.2. Content Clustering Module

4.2.2.1. Item representation

The fundamental idea of content-based RS is using the genuine content of the items and the past preferences of a user to suggest him new items that are similar to the ones he liked before. However, this total dependence on the items' content is a double edged weapon. It allows immediate items recommendation once their content is available. But, it also means that the quality of the recommendations depends directly on items and features that represent them. Thus, it is necessary to give a consistent items' description which contains useful information about them. In fact, the number and the type of items' features-that need to be correctly described- differ from an item to another. Consequently, it is not wise to describe all items by the same, usually small, number of attributes with a known set of values. In order to properly characterize the items, we opt for a textual description providing the necessary information about the items. This helps to better detect correlations between the items and provide precise similarities.

The textual description for items is collected from different information sources and stored in large documents. The problem with the clustering of items, that are described using natural language texts, is the impossibility of directly applying a clustering process on them. This requires the application of an indexing procedure that maps each item into a compact representation of its content. For this purpose, we adopt the most commonly used representation, the Vector Space Model [144], in which the vocabulary consists on unique features (initially terms) describing the items. At first, we performed a preprocessing step in order to switch into a features' space. The first phase in this treatment consists of a cleaning step where we remove special characters and split each sentence into individual words. Then the stop words are removed and the other words are reduced to their stems by using the Porter Algorithm [145]. Finally, the weight of each feature is calculated using the well-known numerical statistic TF-IDF [144]. Hence, the term-frequency vector X_i of the item i is defined as:

$$X_i = \left[tf_{i1} \log\left(\frac{n}{df_{i1}}\right), tf_{i2} \log\left(\frac{n}{df_{i2}}\right), \dots, tf_{iD} \log\left(\frac{n}{df_{iD}}\right)\right]^t \quad \text{(eq.41)}$$

Where tf_{ij} is the frequency of the term j in the item i, df_{ij} is the number of items that contain the term j, n is the total number of items in the collection, and D is the number of terms.

[6] www.google.com
[7] https://en.wikipedia.org/
[8] http://www.rottentomatoes.com/

Such a representation adapts the items for the clustering algorithms, but it generates some irrelevant features that do not contribute to the clustering process and are not beneficial to the similarity measure; on the contrary, they may misguide the clustering process [146, 147]. Therefore, one cannot directly use the clustering algorithms on the items, as it is usually done [148, 149, 150, 151, 152]. Incorporating, in the clustering system, a feature selection method that identifies the relevant features and eliminating the noise caused by non-informative ones is imperative.

4.2.2.2. The Hybrid Feature Selection Method (HFSM)

There are different methods available for selecting either the most statistical or the most semantic informative features. Despite their efficiency, the application of these methods is still insufficient since the first category misses the semantic relationships that may exist between the features that describe the items, while the second category failed capturing the most frequent features. Hence, to benefit from both techniques and perform a complete analysis, it is wiser to combine statistical and semantic treatments. To deal with this issue, we use a clustering algorithm CHFSA [146] that integrated a hybrid statistical and semantic feature selection method HFSM. The use of this technique allows representing the items by selecting only the semantically and statistically relevant features.

The HFSM technique simultaneously selects the most frequent contents by using the CHIR statistic [147] and the most semantic pertinent content-based ones by using the SIM measure [154] through a weighting model. Hence, the term goodness of a feature w is defined by the following formula:

$$HFSM(w) = \lambda * r\chi^2(w) + (1-\lambda) * sim(w) \quad \text{(eq.42)}$$

Where λ is a weighting parameter between 0 and 1.

A term w is considered relevant when the value of its $HFSM$ measure is greater than a predefined threshold.

4.2.2.3. Clustering with Hybrid Feature Selection Algorithm (CHFSA):

The CHFSA algorithm [146] alternates between the clustering process and the selection of relevant features. Therefore, the clustering precision is iteratively improved by the selection of relevant features until obtaining a high clustering accuracy at the stability of the process (when the cluster centroids no longer move).

Using a detailed textual description and a hybrid feature selection method makes the similarity between items more precise and subsequently leads to accurate recommendations. It also helps the clustering process to create compact clusters with similar content. Such approach permits to effectively resolve issues related to item-side. In fact, once the clustering is performed; similarities between each pair of items, using the cosine functions [153], are extracted to build the content similarity matrix. These similarities are used to predict the user's reaction for items where s/he does not provide evidence. Then, when a new item appears in the system, it can be affected to its closest cluster. As a result, a content-based prediction for this new item is done based on its similarities with old items, for which the user has already expressed a preference. The algorithm is given as follows:

CHFSA Algorithm:

Input: S: a set of n items to be clustered
m: number of distinct terms existing in S
k: number of clusters
f: factor in the range of [0,1]
l: number of selected features

Output: Set of item clusters
Set of cluster centroids

1: Perform the k-means algorithm to get initial clusters and centroids
2: **repeat**
3: **for each** of the m features
4: Calculate the hybrid measure HFSM
5: **end for**
6: Rank the terms in a descending order of their criterion function
7: Select the top l features from the sorted list
8: **for each** item in S
9: **for each** feature
10 **if** it is an unselected feature
11: reduce its weight by f
12: **end if**
13: **end for**
14: **end for**
15: **for each** cluster
16 Recalculate its centroid based on the new weights of the features
17:**end for**
18:**for each** item in the new feature space
19: **for each** of the k centroids
20: Compute the cosine measure between them
21: **end for**
22: Assign the item to the cluster that has the closest centroid
23:**end for**
24:**for each** cluster
25: Recalculate its centroid based on the items assigned to it
26: **end for**
27: **until** the centroids no longer move.

4.2.3. Collaborative Module

This module consists on a classical CF approach, allowing the management of users' ratings and the calculation of distances (similarities) between items, in order to be exploited thereafter to generate predictions for them. Thus, this module must access to the original ratings' matrix and compute similarities between items. As we have already mentioned generating predictions is related heavily on the principle of correlation between either users (user-based approach) or items (item-based approach). We used this last variation of the CF approach as we want to resolve the problems related to item-side. The whole data must be represented in a suitable format to make the distance calculation easy later. When it consists in the item-based approach, items must be represented in the users' space, where each item is the m-dimensions vector and each dimension is the rating value given by a user to this item. Thereafter, a distance between two items is calculated relatively to ratings given to the both by common users according to a distance's formula. There are several formulas that can be used. We used the Pearson coefficient, which is accurate:

$$PC(i,j) = \frac{\sum_u (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_u (r_{ui} - \bar{r}_u)^2 \sum_u (r_{uj} - \bar{r}_j)^2}} \quad \text{(eq.43)}$$

- r_{ui} is the rating of the item i given by the user u.
- $\bar{r}_i$ is the mean rating of the item i.

4.2.4. Hybrid Module

Operations performed by the previous two modules were carried out in parallel. While this module is the core of the proposed RS, which is responsible on accessing to the two matrices generated previously and combine them into a single new complete matrix similarity. Then, predictions of non-voted items are computed.

For the hybrid module, two different scenarios are possible. Each one must be treated differently:

1. An old item with only few ratings: in this case the number of ratings is insufficient to make accurate recommendations. Then, this module shall compensate this lack through the use of items' content.
2. A new item with no ratings: in this case computing prediction cannot be rested on ratings. The module would be based *only* on *content*.

Therefore, it consists in alternating between content and collaborative information by making certain balance and achieving an acceptable recommendation according to the situation.

4.2.4.1. Similarity computation

The total similarity between two items is the linear combination of the similarities obtained by the two previous modules:

$$\mathrm{Sim}_{\mathrm{hybrid}} = \mathrm{C}.\,\mathrm{Sim}_{\mathrm{content}} + \ (1-\mathrm{C}).\,\mathrm{Sim}_{\mathrm{CF}} \qquad \text{(eq.44)}$$

Where:

- $\mathrm{Sim}_{\mathrm{content}}$: Items similarity produced by the content clustering module.
- $\mathrm{Sim}_{\mathrm{CF}}$: Items similarity produced by the collaborative module (eq.43).
- $\mathrm{C} \in [0,1]$ Is the coefficient of combination, it defines the contribution of each component in the prediction step.

We must notice that the value of *C* is variable, it changes according to different situations; in fact, depends on the two previous discussed scenarios.

4.2.4.2. Neighbors selection

The computation of recommendations is not limited just on distances' calculation, but also on the selection of neighbors which has a big impact on prediction's quality. So, to identify the most relevant neighbors, we can take into account the following parameters:

- The minimal distance *D* between two elements (threshold).
- The number *K* of neighbors in selection, where we choose the K-nearest neighbors to a given item.
- The minimal number of mutual ratings between elements.

4.2.4.3. Prediction

The prediction's step is summarized by predicting the possible rating's value that a user would give to an item. The prediction's value of an item i according to a user u is computed as follows:

$$\mathrm{prediction}_{\mathrm{ui}} = \overline{\mathrm{r}_{\mathrm{i}}} + \frac{\sum_{\mathrm{j}=1}^{\mathrm{l}}(\mathrm{r}_{\mathrm{uj}} - \overline{\mathrm{r}_{\mathrm{j}}}) * \mathrm{sim}_{\mathrm{ij}}}{\sum_{\mathrm{j}=1}^{\mathrm{l}} |\mathrm{sim}_{\mathrm{ij}}|} \qquad \text{(eq.45)}$$

- l is the size of the neighborhood of the item i.
- sim_{ij} is the hybrid similarity between the item i and his neighbor j computed previously in (eq.44).
- $\overline{r_i}$ and $\overline{r_j}$ are respectively the mean' ratings of the items i and j.

• r_{uj} is the rating of the item j given by the user u.

When a new item appears in the system, the equation (eq.45) becomes invalid as the item is unrated by any user (the value $\overline{r_i}$ can't be computed). In this case, the clustering mechanism is applied solely to affect the new item to its closest cluster C. As a result, the prediction for this item is made by performing a weighted average of its similarities with similar items belonging to its own cluster and for which the user has already expressed a preference. Therefore, we define the prediction in this case as follows:

$$\text{prediction}_{\text{ui}} = \frac{\sum_{\text{j}\in\text{C}} \text{r}_{\text{uj}} * \text{sim}_{\text{ij}}}{\sum_{\text{j}\in\text{C}} |\text{sim}_{\text{ij}}|} \quad \text{(eq.46)}$$

4.3. Evaluation and validation

4.3.1. Dataset

For the evaluation of our proposed algorithm, we perform experiments on movie rating data collected from the Movielens[9] web-based RS. The data set contained 100,000 ratings from 943 users of 1,682 movies; each user in the data set has rated at least 20 items, where rating's value is in 1-to-5 scale.

The items are characterized by their id, title and genre (drama, crime, etc.). In addition, an exhaustive textual content extracted with the Java web scarper, and stored in file data corresponding to each item.

We picked randomly 20% of the whole items to be considered as new items (for which we ignore totally their ratings). The rest of the items (80%) will present the items set $Item_{old}$ on which we learn our clustering algorithm.

In order to evaluate the prediction, we follow the 5-fold cross-validation. Thus, for each item in $Item_{old}$, all ratings are divided as follows training set (80% of the ratings) and a test data set (20% of the ratings). The training set will be used to compute prediction for non-voted items and the test set to evaluate the accuracy of prediction.

4.3.2. Evaluation procedure

In order to ensure the performance of our proposed system, we conducted several experiments under different settings:

4.3.2.1. Non Cold-Start settings

In this case, every user has rated at least 20 items; the mean number of ratings per item is almost 27 ratings in the training set.

Experiment 1: First, we aim to compare the results of the proposed clustering algorithm against baseline item-based CF approach. We evaluate our hybrid approach with borderline weighting cases (C equals respectively to 0 and 1 meaning that either the content-based, based on the proposed clustering method or the CF produces recommendations), in different neighborhood sizes. The evaluation is in terms of MAE and coverage.

Experiment 2: We evaluate our approach by assigning different values to C, to see the impact of varying C on the performance of the system and to find the best value of C.

4.3.2.2. New-item Cold-Start settings

Cold-items can be defined in three different ways [155]: 1) items with only few ratings less than a predefined evidence amount (e.g. Less than 10 ratings), or 2) items with no ratings, or 3) items that exist in the system for less than a certain duration of time. In our case, the dataset doesn't take into consideration the temporal aspect, so we focus on the two first points.

[9] http://grouplens.org/datasets/movielens/

Experiment 3: We aim to identify the relation between the number of ratings per item and the contribution of the proposed content clustering in the recommendation. So, we simulate a new-item cold start scenario by keeping only few ratings 2, 10, 20 or 30 ratings per item. We set the threshold as being the mean number of ratings per item, which equals to 27 ratings.

Experiment 4: Based on the previous experiment, after choosing the adequate value of C we compare our results to SVD-based algorithm presented in [156] which has proven its efficiency in cold-start situations.

4.4.Results and discussion

The conceived hybrid system aims to improve the recommendation quality, as performing well in new-item cold-start situations. Since the proposed system relies primarily on the combination of the content clustering and CF modules, we varied the coefficient C according to the different possible values ($\boldsymbol{C} \in [\mathbf{0}, \mathbf{1}]$), until determining the optimal value of C for which the system gives better results in term of different evaluation metrics.

Experiment 1:

Initially, we compared the behavior of RS when it is based on only either content (C=1) or CF (C=0). We vary also the number of neighbors taken into account for each item. The prediction is evaluated according to MAE and coverage and the obtained results are reported in Figure 21.

From the left curve in Figure 21, we observe that the pure content algorithm outperforms the CF and shows good results in term of accuracy (MAE doesn't exceed 0.1), unlike classical content approaches that ensure a low accuracy. We also notice the impact of the number of neighbors considered on prediction accuracy, as it mustn't be too small neither too big. The high coverage in figure 23 shows the algorithm's ability to compute the prediction for any item regardless of the number of ratings, due to the complete similarity matrix generated by the content clustering based on Hybrid Features Selection Algorithm.

The initial results prove the power of our hybrid clustering to compute accurate similarities between items resting only on a large content which leads to a high quality prediction. Since these similarities are computed once the item's content is present, it helps to construct the fullest similarity matrix as possible, which solves the sparsity problem. It is also proven by a very high coverage. On the other hand, CF doesn't perform well, in most cases it seems unable to calculate prediction, or they are less accurate. This is due to the sparsity of the matrix (94% sparse), because in CF the more we have ratings, the more accurate predictions are generated.

Experiment 2:

Figure 22 shows the variation of MAE and coverage, according to the constant C. We may notice easily that the system tends to give better results when C is bigger (i.e. the contribution of content module is more important); such a result illuminates the following:

- Sparse matrices lead to inaccuracy of RS
- More we lack sufficient ratings, more content is encouraged and CF is neglected.
- Our proposed content clustering delivers a good prediction even with few ratings, unlike the CF approach.

Experiment 3:

As the Hybrid RS based on CHFSA is conceived to improve the performance of RS under new-item situations, we simulate a new-item cold-start scenario by keeping only few ratings K per item (K=2, 5, 10, 30). Then, we conduct several experiments by changing the value of Coefficient C to find out the relation between the number of ratings per item and the contribution of each of the two components (our proposed CB recommendation using CHFSA and the classic item-based CF) in prediction computation. The obtained results are presented in Figure 23; each curve shows a particular behavior, according to different cold-start situations.

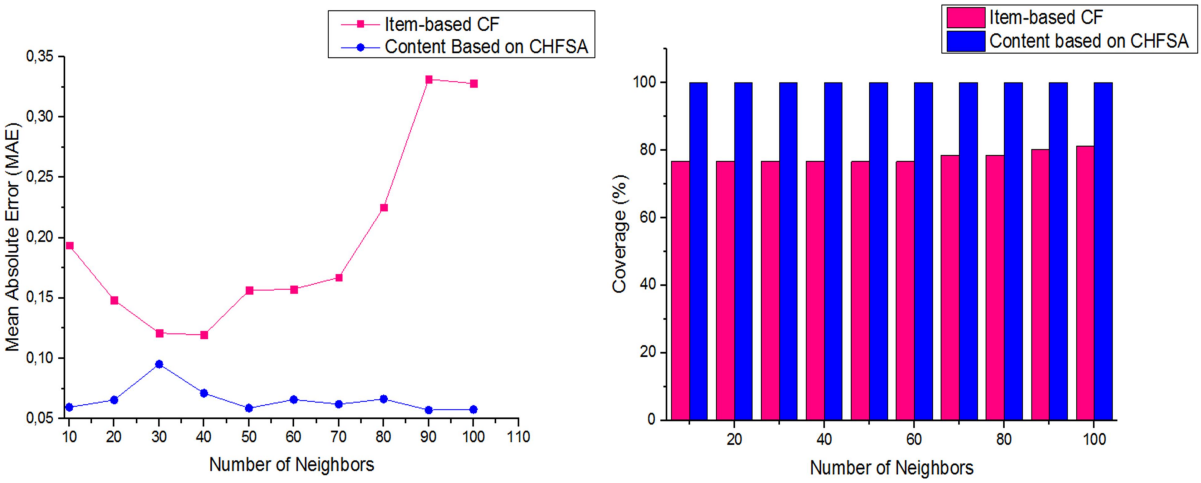


Figure 21: Comparison of MAE and coverage between Item-based CF and Content based on CHFSA.

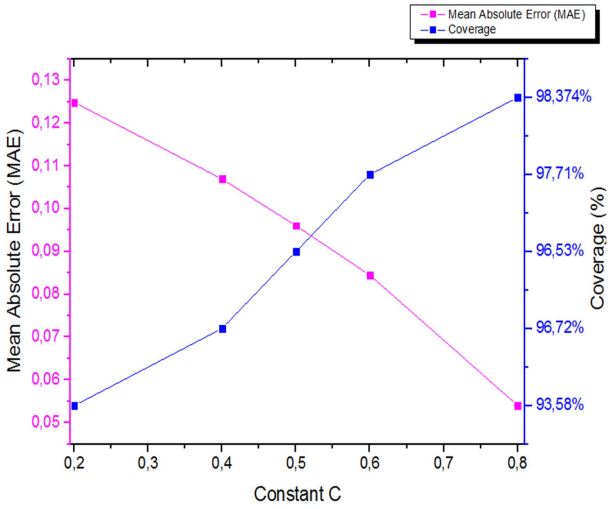


Figure 22: Variation of MAE and Coverage according to C.

The Figure 23.a shows an extreme new-item cold-start situation, in which the number of ratings K is very small, the MAE reaches a surprising value (over than 1) when C equals to 0. This proves that CF can't perform better in such cases. On the other side, along with the increasing value of C, the MAE drops until arriving to a lower value when C=1 (our content algorithm is dominant).

The Figure 23.b shows a different shape than the first one, the lowest value of MAE is reached in C=0.8. Then, the CF tends to take more importance when the number of ratings K has increased. Figure 23.c shows a little change compared to Figure 23.b. Even if the number of ratings K has increased, the contribution of content based on CHF1SA is emphasized at the expense of CF, the best value of MAE is reached when C=0.7. The value K=10 remains negligible compared to the total number of users and items. The last curve in Figure 23.d, emphasizes the importance of CF as the best MAE value is obtained at C=0.3. This was due to the number of ratings per items which is relatively important (K=30), as it is bigger than the mean number of ratings per item (equals to 27).

Generally, the change in shape over the three curves confirms that:

- More items lack ratings, more our Clustering algorithm based on HFSM has importance and its contribution tends to be bigger.

- The bigger the number of ratings K is, the smaller become our contribution in favor of CF approach.

This just indicates that our proposed method has a definite advantage in such new-item cold-start cases, since it improves the prediction in such situations. The proposed algorithm based on clustering with HFSM is a good alternative when the classical CF approach fails to deliver good predictions.

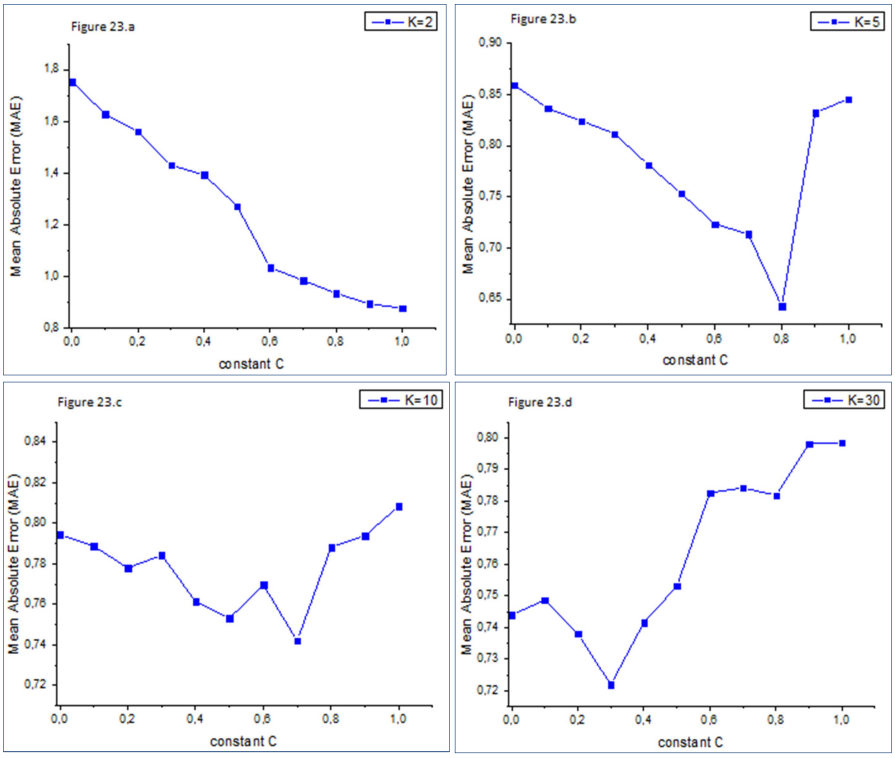


Figure 23: Artificial new item with only few ratings K equals respectively to 2, 5, 10 and 30.

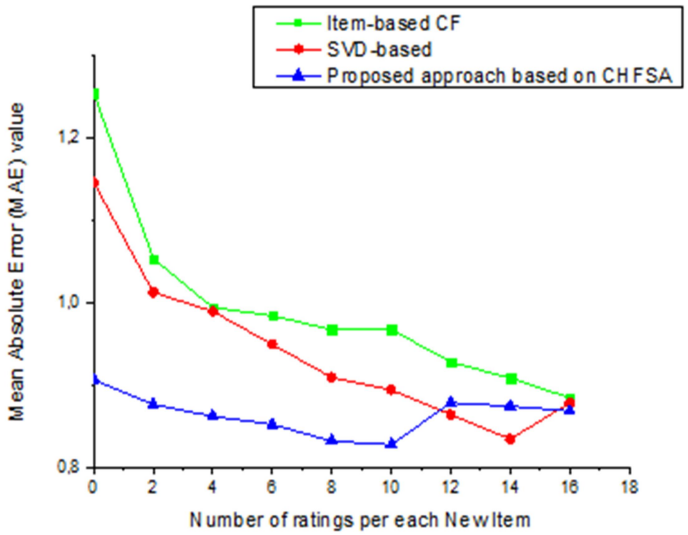


Figure 24: Comparison of the proposed method against baseline approaches in new-item settings.

Experiment 4:
According to results obtained in the experiment 3, we vary the value of C between [0.3, 0.7] until finding the best value of C (the mean number of ratings M=27 and 10<M<30). The best value of MAE is reached in C=0.6. After setting the value of C, we evaluate the hybrid approach against the item-based CF and SVD-based approaches in term of accuracy. We simulate the new-item scenario by changing the number of ratings per each item (See Figure 24).

4.5.Conclusion and perspectives

Incorporating content into CF recommenders, can improve significantly the accuracy of the system as it overcomes cold-start situations when we lack of sufficient ratings. However, the CF was greatly favored over the CB approach as this last ensures generally a low accuracy; we presented a hybrid RS based on a powerful content clustering algorithm using Hybrid Features Selection Algorithm (CHFSA), which maximizes the profit gotten from content and upgrade RS's performance.

The proposed system overcomes the weaknesses of content and CF, by consolidating one through the other. The conceived RS aim mainly to tackle problems related to item-side; we conducted many experiments which simulate new-item situations in which the system worked great. The proposed solution may be adopted in the context of many recommenders such: news, web pages, books, articles, etc.; in which an item has an important content.

Last but not least, our proposition is paving the way for further issues to be thoroughly addressed in future research works, namely, adapting our approach to support the use of multiple data sources and automatically switching from one to another. Another interesting topic that can be addressed is the scalability and the reactivity of a recommender system.

5

A SCALABLE RECOMMENDATION ENGINE ON LARGE SCALE

The quality of recommendations is largely determined by the quantity and quality of data. "Garbage in, garbage out," has never been more true than here. Having high-quality data is a good thing, and generally, having lots of it is also good. […]

[…]The recommender ought to be both fast and produce good recommendations. Of those two, it's better to focus on producing good recommendations first, and then look to performance. After all, what's the use in producing bad answers quickly?

Owen, Sean, and Sean Owen. "Mahout in action." (2012).

SUMMARY

A SCALABLE RECOMMENDATION ENGINE ON LARGE SCALE 94
5.1. Introduction 95
5.2. How RSs meet Big Data challenges? 96
5.2.1. Big Data characteristics 96
5.2.2. RSs vs. Big Data 96
5.3. Related Works 98
5.4. The Proposed Distributed Recommendation Engine 100
5.4.1. About the technology 100
5.4.2. The proposed Recommendation Engine architecture 102
5.5. Experimental Results and Performance Tests 104
5.5.1. Experimental Environment 104
5.5.2. Evaluation Criteria 104
5.5.3. Experimental Results 105
5.6. Conclusion 110

5.1. Introduction

During this thesis, our aim consists not only in implementing and presenting more accurate recommendations but also to deal with the large-scale amount of data. Indeed, the two previous chapters illustrated our conducted works and present our proposed strategies to solve the cold-start problem that affects the recommendations quality (i.e. Prediction accuracy and coverage) and prevents RSs to perform more efficiently. However, even though the proposed algorithms have proved their efficacy in term of accuracy, their overall runtime performance degrades dramatically (i.e. the time needed to produce new recommendations is significantly higher) when they are implemented over large-scale where numbers of users and items grow increasingly. In fact, this phenomenon is known as scalability problem that is commonly faced by RSs, especially those using CF methods [158]. The current chapter aims to complement previously done researches, by improving the scalability of the implemented proposed algorithms, which consists of applying a scale-out approach or adding computer nodes to run the proposed recommendation engine.

Generally, RSs are considered greedy by nature since their accuracy depends merely on the amount of)information they used to compute predictions. As a consequence, such systems are highly computationally intensive and thus they suffer from scalability issues related to data storage and recommendations computation in a timely manner [159]. Lately, the world has entered a big data era and the amount of accessible data flowing on the web increases exponentially day after another. New challenges are faced and it became very difficult to generate recommendations *accurately* and *quickly.* Hence, traditional recommendation techniques became obsolete and unable to perform well when processing or analyzing data on a large scale. As a result, they need to be re-evaluated to meet with new challenges. Henceforth, a robust RS has to remain efficient and offers *millions* of *real-time high-quality* recommendations in *few seconds* for the increasing number of users and items.

Handling the scalability issue in recommendation systems can be ensured by many ways. The most used solution consists of utilizing an offline approach in which the recommendations for each user in the dataset are computed all at once, in a batch computation. Nevertheless, this solution is not practical enough since it doesn't ensure a real time computation and subsequently it is inappropriate for low latency operations. In addition, using a simple relational database (RDB) or a set of clever scripts may be also used to process data more speedily. However, data tend to grow and it may outgrow the used RDB or the scripts rather quickly. On the other hand, supporting parallel and distributed computing, which is at the heart of Big Data is considered as the most fashionable way to scale heavy processing. For this reason, traditional recommendation paradigms have to be adapted to meet new technologies invented in Big Data (e.g. Apache Mahout, Map Reduce of Google, Apache Hadoop of Yahoo! And Apache Spark from University of California, Berkeley [160] and so on).

In our context, we introduce an algorithmic framework built on the top of Apache Spark- a newly introduced distributed Big Data processing framework- for a parallel computing, in order to simulate how professional recommendation engine would work in a minimized scale. Spark allows our algorithms to scale and to perform efficiently and accurately, since the Resilient Distributed Datasets (RDDs) provide a faster way for iterative and interactive algorithms to execute as compared to the traditional Map-Reduce paradigm. The aim of this chapter is to demonstrate the viability of our proposal in a large-scale industrial context. The viability of the approach is assessed both in terms of the quality of the recommendations and in terms of operational efficiency, considering in particular the computation time and the necessary infrastructure for the implementation of the model.

The rest of this chapter is worded as follows: the section 5.2 is dedicated to remind the Big Data characteristics represented by the different Vs as well as to discuss the new challenges posed by RSs actually to response subsequently to the question why the use of Big Data techniques is suitable for RSs. The section 5.3 discusses the related works done in the regard of RSs on the context of Big Data. The

proposed distributed recommendation engine is presented in the section 5.4. Finally, the experimental environment, the conducted performance tests and the obtained results are covered in the last section.

5.2.How RSs meet Big Data challenges?

5.2.1. Big Data characteristics

Big Data and its analysis are at the center of modern science and business. These data are generated from online transactions, emails, videos, audios, images, click streams, logs, posts, search queries, health records, social networking interactions, science data, sensors and mobile phones and their applications [161]. Their massive growth makes them difficult to capture, form, store, manage, share, analyse and visualize via typical database software tools.

In fact, the traditional techniques are not suitable to cope with the massive amount of data (Volume) that is generated from heterogeneous sources with diverse formats (Variety) and with unprecedented velocity. For these reasons, Big Data requires a revolutionary transition from the traditional data analysis, characterized by these six following features - called also dimensions, features or challenges:

- **Volume:** The most visible feature of Big Data, it refers to the increased amount of data that is exponentially created each year (scale from Terabyte to Zetabyte) and also the growth of data sources even for a single domain. The main issue here is how to deal with excessive volume of data and reduce the storage dimension.
- **Variety:** As a direct consequence of the deployment of new sensors and the multiplication of data sources, data can be generated by users and machines in any format and new data types are created; it includes structured and unstructured data of all varieties (text, audio, video, log files…). Thus, we need to deal with data in multiple formats coming from heterogeneous sources.
- **Velocity:** This feature is time sensitive and refers to the real-time demand of data. Velocity signifies the frequency at which data is generated and the speed at which data are analysed.
- **Veracity:** Even in the same domain, we may encounter different data qualities. This dimension refers to the uncertainty due to data inconsistency and ambiguities. It allows guaranteeing the reality and credibility of the data.
- **Variability:** Among the factors that often cause problems for data scientists. It refers to the inconsistency and the various meanings of data depending on their context.
- **Value:** There is good information hidden in Big Data, the issue here is how to identify what is valuable and transform it for analysis.

5.2.2. RSs vs. Big Data

Recommendation systems represent one of the most important issues in the data mining. In fact, data mining techniques are very hungry for data, hence the more we acquire data, the more precise results we get. The recommendation process is mainly carried out according to two main steps: 1) Read a huge amount of data that maps a user with some preferences for an item 2) Find an item that should be suggested to the user.

There are many reasons that make Big Data techniques very suitable for recommendation processes. One of these reasons is the variety of options provided to users, whose attention span is very limited, as a result an efficient RS have to deal with this variety and to sift the most relevant items among the whole of the dataset. Another reason is that larger datasets of items, user behaviour, and related data are required during the process to produce relevant recommendations. Furthermore, the variety of recommendations use cases makes them fast changing and complex, hence the multitude of recommendation paradigms. Therefore, to meet these new requirements posed in RSs, an agile, flexible, scalable and end-to-end platform seems to

be the right solution. The requirements and different problems confronted in RSs projected on Big Data features (Vs) as well as special use cases are depicted in Table 10.

Table 10: Big Data features Vs RS's requirements.

Big Data characteristics (6Vs)	***Recommendation Systems requirements: why RSs are affected by the Big Data V (Special use cases and examples)***	***The concerned RS's technique***
Volume	**Problem:** A RS is unable to make good recommendation for new users or to recommend new items. **Solution:** The *more data* acquired about a user, the more his profile *is rich*. The RS knows the user better and is able to offer him good recommendations. The same for an item, the *more* we have *feedbacks* about it (ratings, comments, rich content data, etc.…), the better it is recommended for adequate users.	All RS's types are affected by this feature: (CF, CB, Hybrid)
Variety	**Problem:** The data sparsity already discussed, paralyze the RS (difficulty to recommend new-items and serve new-users). **Solution:** To face this issue, new techniques were used to compensate the lack of data by embedding new data sources to work together in order to improve the recommendation's quality. New generation RSs act on different types of data at the same time (e.g. Demographic data, ratings, navigation history, external information in social networks, etc.…)	Hybrid RSs
Velocity	**Problem:** Users are very demanding and hard to please; hence RSs have to capture their attention from the first use by offering them high quality recommendations. A high-quality RS refers to a system which is able to understand the real users' tastes, and offer them in turn items that may be interesting and serendipitous for them. Users are also hurry to get what they want, so the RS must satisfy them very quickly. **Solution:** Changes in users' profiles and new feedbacks on items must be taken into account, hence the updates must be conducted rapidly. All these requirements prove that the RS must act quickly with the high speed of changes in data and analyze it in real time manner.	All RS's types are affected by this feature: (CF, CB, Hybrid)
Veracity	**Problem:** The real identity of the user must be checked before saving any feedback from him (it may be another person who uses the user's account temporally). RSs that base their analysis on implicit feedback, must interpret feedback very carefully. For example, if a RS save the time spent by a user on a page or the number of times that the user has listened to a certain song, but in reality the user has left his session open and quit. **Solution:** A RS has to tackle the inconsistency of data by eliminating any ambiguities that can be generated.	All RS's types are affected by this feature: (CF, CB, Hybrid)
Variability	**Problem:** In real cases, there is no valid recommendation for a long term. Since users' preferences change during the time, or vary from one situation to another. For instance, the type of movies to watch depends on the person with whom you will watch it. If you are with your parents or children you may prefer a comedy or drama, but if you are with a conjoint you will prefer romance. Also, the style of music to listen depends on the user's mood Another example for news recommendations, the same user may prefer to consult new about weather, stock markets in the morning. On the afternoon Political events and news may be preferred. In the evening, when the user is exhausted after a long working day he seeks for light news like celebrity news and gossip.	Context-based & Sentiment-analysis based recommendations

	Solution: this variability of user data that is reached by the changing context and users' moods and situations must be taken into account	
Value	**Problem:** - Modern recommendation systems mustn't rest their processing on basic data such as ratings, items' features, and so forth. **Solution:** But rather, RS must transform simple data to capital information that may be hidden and benefit from it. - For instance, recommendations can be rested on implicit feedbacks from the user. - Also, user's information can be extracted from social networks, cookies, etc.	Hybrid recommendation

5.3.Related Works

Lately, the emergence of Big Data motivates the researchers (data scientists) to conduct more exhaustive researches that aim to develop appropriate and efficient strategies to build reliable RSs while benefiting from the advantages of new Big Data technologies. Among these researches, we find [162], in which authors gave a close-up on Big Data analysis that both allows a good understanding of the field and reports the emerged opportunities and challenges in the Big Data environment. Similarly, the conducted study in [163] aims to describe the components of modern RSs in conjunction with the features and challenges brought by Big Data on the recommendation field.

In fact, developing a real life recommendation engine, whether for commercial or personal ends, becomes a complex and challenging task, alongside the new requirements in term of high quality and real-time response. Indeed, there are many alternatives to support recommendation applications development, but the choice between them is a crucial primitive stage. Particularly, the development of modern RSs can be carried out in many different ways: 1) the platform is developed from the scratch 2) an existing recommender engine may be contracted (e.g. OracleAS Personalization) or 3) code libraries can be adapted, or a platform may be selected and tailored to suit (e.g. LensKit, MymediaLite, Apache Mahout, etc.). In the most cases, a combination of these approaches is employed to overcome the raised challenges of modern approaches on the recommendation field Generally, scalable Big Data frameworks are needed, to support storage and analysis in RSs due to the massive amount and the high velocity of processed data [164, 165].

In particular, despite the big success achieved in the few recent years by CF techniques, they still suffer from many problems when they are implemented on a large scale. More specifically, their data-intensive nature raises highly non-trivial issues related to computational efficiency and scalability. Ordinarily, even if this computation can be performed on a single machine, it remains impractical since it exposes the system to the breakdown risks whenever that machine fails [166]. Furthermore, the computational requirements of the system often surpass the performance of the used machine, which makes the use of one machine disadvantageous from a scalability and cost standpoints.

Under those circumstances, many pieces of research have been done previously to solve the scalability problem faced by traditional CF techniques. Some of them were focused on improving the CF algorithm itself by investigating the usage of several existing similarity metrics [167]. Notably, similarity computation is carried out offline to build the item-to-item (resp. user-to-user) similarity matrix, to be exploited later on the recommendations production step [34, 135]. At the same time, other works have been performed on the Big Data analysis field, especially, for RSs through using different techniques and platforms [168]. The most of them were dedicated to using Apache Mahout, as an extensible open-source data mining library in the context of recommendation platforms. Having many real-life applications (Facebook, LinkedIn, Twitter, Yahoo, Foursquare), Apache Mahout provides a rich and varied recommendation functions and supports two different processing modes: 1) a non-distributed mode that provides an exhaustive set of components allowing the construction of a customized RS algorithms and 2)

a distributed processing mode that runs on the top of Hadoop infrastructure and perfectly manages a huge amount of data thanks to the scalable Map-Reduce programming paradigm. Indeed, Mahout became increasingly a popular choice for organizations seeking to take advantage of the large scale Machine Learning techniques [169].

Lately, other growing researches on large-scale CF methods aim to benefit from recent advances in data processing technology, which consists in the use of new emerging tools and platforms for distributed and parallel data processing, namely Apache Hadoop [170, 171, 172]. As a popular distributed computing framework, Hadoop includes two main components responsible of its success: 1) the Hadoop Distributed File System (HDFS): for data storage, and 2) an implementation of the Map-Reduce programming paradigm designed for data processing and computation. Opting for the use of such fashionable technology ensures the splitting of data into several partitions across many affordably small machines of the cluster. Therefore, the computation is executed on each machine in the cluster according to the Map-Reduce paradigm (i.e. Map and Reduce operations) which consequently improves the speed, especially when the number of nodes attached to the cluster is bigger. Another key point of the Hadoop ecosystem relies on the high availability of the HDFS files through data replication mechanisms over multiple nodes (machines), which doesn't only speed up the calculation step, but it resolves also downtime risks of failed machines and brought highly fault-tolerance aspects. Moreover, whenever more speed is required additional machines may be added to the cluster, which offers a seamless scalability. In the case of necessary resources lack, the resources of many machines are readily available these days through cloud computing providers such as Amazon's EC2 service (http://aws.amazon.com). These machines can be rented temporarily on demand and used within the cluster, which eliminates the upfront cost of implementing large-scale, data intensive workflows.

Although Large-scale recommendation algorithms can extremely benefit from the fault tolerance and high scalability using Hadoop, their structures must be completely changed according to the Map-Reduce programming Model. In other words, the user has to specify: 1) the map function that processes key/value pair in order to generate the intermediate key-value pairs and 2) the reduce function that merges all intermediate values associated with the same intermediate key. Then, the underlying parallel processing platform uses a distributed file system to provide high throughput access to the data and manages the horizontal scalability of adding more machines to the cluster and dealing with machine failures. Moreover, the underlying parallel processing platform uses HDFS to provide high throughput access to data, but cannot cache the intermediate data in memory for a further requirement and need to write it to hard disk, which extremely diminishes the performance of the system. In addition, Hadoop paradigm doesn't fit fully with iterative processing like CF algorithms and does not support cyclic data flow because it needs to repeatedly access data from the disk which makes it slower [158].

Actually, new computing platforms like Apache Spark are getting prominent in the field of Big Data analysis that tries to fill the gaps on Hadoop/Map-Reduce since it can handle in-memory data processing. Earlier experimentations have shown that CF algorithm run on Spark is faster and have better scalability than the same algorithm ran on Hadoop Map-Reduce [173], as well as the computation time is more speeder when more nodes are attached to the cluster [166]. Notably, Apache Spark beats Hadoop owing to many reasons: 1) Spark is generally a lot faster and speedier than MapReduce because of the way it processes data. In-memory processing is faster since no time is spent in moving the data/processes in and out of the hard disk. For that, Spark caches much of the input data on memory for further exploitations which improves the performance of iterative algorithms that access the same data repeatedly. While MapReduce operates in steps, Spark operates on the whole data set at once. Notably, *"Spark can be as much as 10 times faster than MapReduce for batch processing and up to 100 times faster for in-memory analytics"* as said Kirk Borne, principal data scientist at Booz Allen Hamilton. 2) Spark suits to do analytics on streaming data since it emphasizes on the velocity of the data and processes it within a small period of time. Common applications for Spark include real-time marketing campaigns, online product

recommendations, cyber security analytics and machine log monitoring. Indeed, in large-scale context recommendation algorithms need to be improved to get better results, so they can be performed on such platforms for real time and efficient recommendations. For recent works done in this regard; the reader can be referred to [174] and [175]. Accordingly, in our context, we used the Spark framework in a cluster, in order to implement a distributed version of our recommendation algorithms proposed earlier to gain in terms of scalability and response time. The next section is dedicated to presenting the architecture of the proposed recommendation engine in details.

5.4.The Proposed Distributed Recommendation Engine

In this section, we propose a distributed implementation of our proposed recommendation algorithms over the Apache Spark platform for large-scale data processing. First, we introduce the used technology. Second, the architecture of the implemented recommendation engine is presented. Then, the main steps of recommendation process conducted are illustrated and prominent algorithms of each step are introduced.

5.4.1. About the technology

This section is dedicated to introduce the Apache Spark framework which enables distributed processing and ensures high scalability. We will investigate the main commonalities and differences between Spark and Hadoop Map-Reduce framework, and then we discuss the strengths behind the success of Spark.

Apache Spark

Apache Spark is an open source, fast and distributed data processing framework started in 2009 by Matei Zaharia at UC Berkeley. Indeed, Spark was developed especially to respond to the limitations of Hadoop's Map-Reduce at processing and performance levels. In fact, Map-Reduce Paradigm is regarded as inappropriate in two different scenarios: 1) iterative algorithms where iterations are expressed as a MapReduce jobs and data have to be loaded at each iteration from the disk and 2) interactive Big Data analysis applications where each query is a separate job that has to load data from the disk instead of loading it at once and use it repeatedly. This persistence of intermediate data to disk between Map and Reduce steps presents a major flaw and causes a drop down of global performance. In contrast, given its advanced in-memory computing capabilities, Spark operates faster than disk-Based implementations of Hadoop [176]. In fact, Spark was founded to fill the gaps of Hadoop through loading the whole data into the memory once, to be subsequently used as many times as necessary. Furthermore, Spark caches also the output of each parallel operation in the memory of each cluster node instead of writing it on the hard disk as Hadoop do, which allows a direct access to data without the need of a reading file system at each iteration of the algorithm. The real time and in-memory data processing make Spark well suited to do analytics on streaming data, like from sensors on a factor floor (i.e. Interactive Big Data analysis), or have applications that require multiple operations (e.g. Iterative machine learning algorithms). In our context, the use of Spark will improve significantly the AL and recommendation steps, since they are based on the interactivity with the user and need to be conducted in real-time. Notably, the user is requested to give his feedback about items to build his profile, then the recommendations have to be generated very quickly and displayed to the user in question.

Spark Programming Model

The Figure 25 illustrates the dataflow programming model using Spark, where the implementation of a parallel program is conducted using a driver program running on the master node of Spark cluster, that is responsible of controlling the flow. In fact, a driver program is expressed as a sequence of parallel operations (i.e. Transformations and actions) defined by Spark's user as required and executed separately on each slave node (called also a worker node) on the cluster. Spark framework provides a panoply of core abstractions to accomplish the dataflow programming model mentioned above, most notably *Resilient Distributed Datasets (RDD).*

The concept of *RDDs* is the core of Spark, which is the main abstraction allowing working with data. RDDs are a read-only collection of records (data) that is distributed across many nodes into partitions,

which allows re-building data if one of the partitions is lost. In fact, any work in Spark application is focused on manipulating RDDs (creating a new RDD, Transforming an existing RDD or executing operations/actions on an existing RDD to compute a result). The in-memory cluster computing presents the main feature of spark, the elements of an RDD need not to exist physically to be treated which ensures a fast data sharing and an increase of the application's speed.

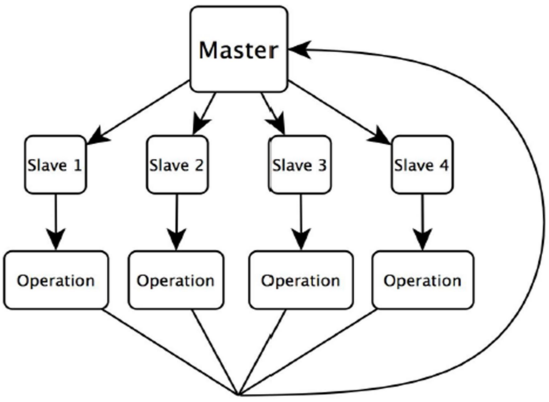


Figure 25: Data Flow in Apache Spark.

Spark allows to its programmers creating RDDs on two different ways, either by loading external data (File text, DB, HDFS, etc…) or by parallelizing a collection of objects (like an array, List, etc…). Furthermore, Spark supports two different kinds of *parallel operations*: 1) transformations that are operations on an RDD that return a another new RDD, or 2) actions are operations that return a final result to the driver program or write it to an external storage system. In fact, the main characteristic of RDDs that makes their success is their lazy and ephemeral nature and transformations are subsequently evaluated lazily. This means that the data in each partition isn't available until applying *parallel operations* (actions) to it and after the use data is eliminated. Under certain circumstances, the data is supposed to be re-used later on the algorithm, so it can be persisted by using the Spark's cache action (i.e. Persist or cache actions). This latter allows Spark to keep the RDD in question in memory of worker nodes in order to improve the performance of the algorithm.

Spark Stack

Spark manages a stack of high-level tools (libraries) that can be combined in the same application seamlessly (see Figure 26):

- *Spark Core:* contains the basic functionalities of Spark. It provides task scheduling, memory management, fault recovery and others.
- *Spark SQL:* represents a component built on the top of Spark Core that introduces a data abstraction called DataFrames. It allows operating in structured and semi-structured data as well as querying this data via SQL or HQL (Hive Query Language).
- *Spark MLlib:* is a package that contains the most common Machine Learning (ML) functionalities such as clustering (K-means and LDA), Matrix Factorization, CF and so on.
- *Spark GraphX:* consists in a library for graph manipulation.
- *Spark Streaming:* is a Spark component that enables processing of live streams of data, it leverages Spark Core's fast scheduling capability to perform streaming analytics.

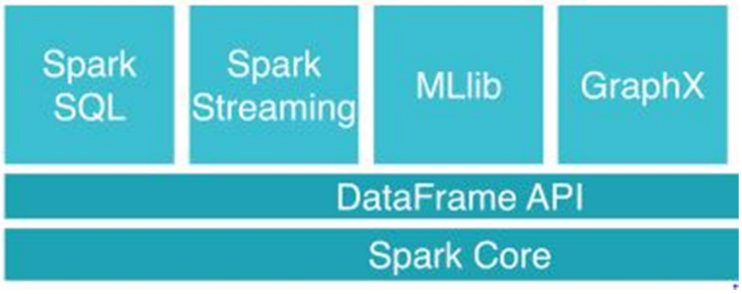


Figure 26: Apache Spark Stack.

5.4.2. The proposed Recommendation Engine architecture

5.4.2.1. High-level architecture

Our recommendation engine is made up of three main modules (Figure 27):

1) *Extraction, Transformation & Load (ETL):* this module is responsible of extracting reliable information (ratings, items' metadata, users' information, etc.) from data sources that can be in different forms (text files, CSV, MySQL, etc.). First, all data is extracted from different sources, then the whole of the data is cleaned by removing users without any ratings or those having less than 20 ratings in their profiles.
2) *Multi-Criteria Active Learning (AL):* this module computes items' weights based on the CCSDW method proposed in chapter 3. Then, it ranks these items to new users in order to get back their ratings on them and construct users' profiles. More precisely, this module is responsible for constructing and updating the new user profile, by interacting with the user in question and requesting him to give his feedback about a well selected set of items using CCSDw method. Thanks to the use of Spark, the functionalities of this module are ensured rapidly, which allows to build an accurate user profile in order to offer him recommendations and even any changes in the user's preferences will be taken into account to improve continuously the quality of generated recommendations.
3) *Multi-Criteria Recommendation:* this module acts on data gathered from the previous module and stored in MYSQL Database to deliver recommendations using Multi-Criteria ratings.

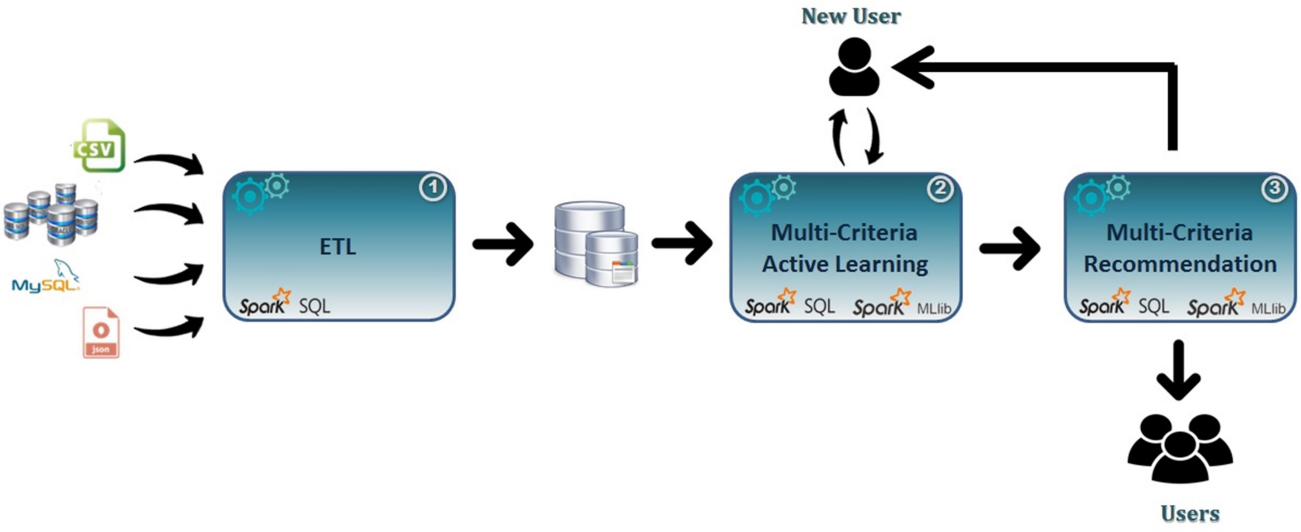


Figure 27: High-Level architecture of the proposed Recommendation Engine.

In order to implement the algorithms (functionalities) of these modules we required the integration of Spark SQL and Spark MLlib components in addition to Spark Core for low latency computing. The figure 27 shows the high-level architecture of the proposed recommendation engine.

5.4.2.2. Components of Recommendation engine

Each of the presented modules uses different types of inputs and performs many actions. The general parallelized workflow is represented in figure 28, which shows the different steps carried out by each of the implemented modules. The first module of ETL acts on raw data to make the cleaning and loading of data into the database, the outputs of this module are users, items and ratings tables. Then, the generated tables are used subsequently by the two other modules.

Moreover, the second module of CCSDW operates on ratings matrix represented by the ratings table. It performs different parallel actions that aim to create decision matrix, computations of the CCs and SDs of available criteria, computation of criteria's weights and finally the weighing of items.

Concerning the third module, it uses the whole of tables generated by the first module and the criteria's weights that were computed by the second module to make personalized multi-criteria recommendations. To this end, many tasks are performed: 1) the creation of the individual ratings matrix for each criteria 2) based on each individual rating matrix individual pairwise items' similarities matrix are computed 3) construct the global similarity matrix based on all of the individual similarity matrices 4) computing the item's prediction for un-rated items and 5) the generation of personalized recommendations as Top-N recommendations for a given user.

N.B. Each of the separate tasks presented of each module is implemented as a separate distributed function using Spark.

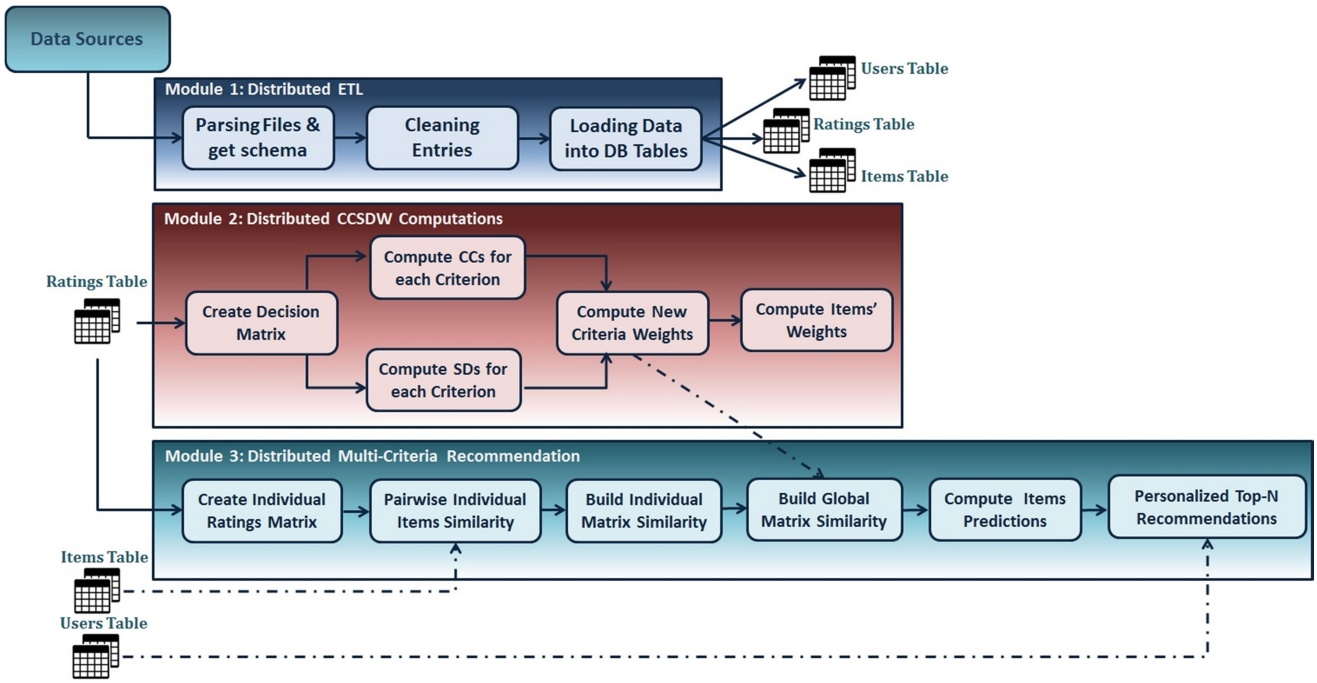


Figure 28: Parallelized Recommendation Engine: Workflow Diagram on Apache Spark.

5.5. Experimental Results and Performance Tests

5.5.1. Experimental Environment

In this section, we present the results of the experimental evaluation of our parallel algorithms on two multi-criteria large data sets: Yahoo!Movie (YM) and HRS (see their descriptions in section 3.4.1.). We first evaluate the parallel algorithms of each module of the recommendation engine on a local machine running Spark 2.1.0 with one 4-core CPU and 6 GB of memory. Next, we run the same experimentations on a cluster of 10 slave machines and one master machine. Each cluster node is equipped with 3.4 GHz Intel(R) Core i3(R) with 4 GB memory, Java 1.8 and Ubuntu 14.04 LTS.

5.5.2. Evaluation Criteria

5.5.2.1. Running Time

Running time is the time needed by the algorithm to generate results. The running time is measured for different Apache Spark cluster size.

5.5.2.2. Speedup

Speedup is the running time of the program ran on smallest cluster divided by running time of the program ran on another cluster which has more number of nodes. The calculation of speedup is as follows:

$$S_p = \frac{T_a}{T_p} \qquad \text{(eq.47)}$$

In this equation, S_p denotes speedup of the recommendation algorithm on a cluster with p nodes, T_a is the average running time of a program ran on the smallest cluster that contains a nodes (processors or cores), and T_p is the average running time of a program ran on the cluster with p nodes.

An ideal speedup is increased linearly with increasing number of nodes used, i.e. $S_p = p$.

5.5.2.3. Parallel Efficiency

This parallel efficiency measurement indicates how efficient an application is when using increasing numbers of parallel processing elements (CPUs, cores, nodes, etc.). In other words, it measures the fraction of time for which a processor is usefully utilized (i.e the speedup per processor). It is commonly defined as the speedup divided by the number of units of execution.

$$E_p = \frac{S_p}{p} \qquad \text{(eq.48)}$$

In this equation, E_p denotes parallel efficiency of a given algorithm, S_p is the speedup of the algorithm and p is the number of the parallel processing elements.

The ideal efficiency is obtained when the speedup is ideal, i.e. E_p= 1=100%.

5.5.2.4. Scalability

Scalability is a measure of a parallel system's capacity to increase speedup in proportion to the number of processors.

Scalability is the quality of the program to remain efficient for a large number of processors and / or cores. The scalability of a program is penalized by:

1. The overhead due to communications.
2. The more or less balanced distribution of tasks between processors and / or cores (Load-Balancing).
3. The fraction of parallel code (Amdhal's law)

5.5.2.5. Amdahl's Law (1967)

The speedup of a program using multiple processors in parallel computing is limited by the time needed for the serial fraction of the problem. The speedup is limited by the non-parallelized part of the code. Amdahl's Law implies that parallel computing is only useful when the number of processors is small, or when the problem is perfectly parallel.

5.5.3. Experimental Results

Since the implemented multi-criteria recommendation engine includes different modules, each of them performs separate tasks, we also evaluated them separately in terms of the various evaluation criteria previously presented, in order to demonstrate the general system's scalability. For that, we ran our recommendation engine using Spark and we measured the runtime as we increased the number of data partitions α that Spark uses to distribute the work across, given two different settings: Pseudo-distributed and Fully-distributed.

NB: It is worth to note that the first implemented algorithms (sequential program) were dramatically too slow. The treatments were launched in batches and ran on a single node, as a consequence the overall response time was over than 100 hours.

5.5.3.1. Pseudo-distributed

Under these settings, a single local machine is used which incorporates multiple cores as a standalone Spark cluster. In this case, α represents the total number of available cores in the machine. In order to demonstrate the scalability, the different algorithms of each module are run on a cluster with one node (core) to produce results, and the overall running time (the running time of the whole algorithm of each module) was recorded. Then, the same experiment is also executed on other clusters which contains two to six nodes (cores). The Speedup is calculated by comparing the running time through different cluster size using the equation 40, where the smallest cluster contains one single core. The different obtained results are shown in Tables 11, 12, 13 corresponding respectively to the performance of ETL, AL and Recommendation modules.

Table 11: Performance test for ETL Module under pseudo-distributed settings.

Number of Cores α	**Running Time (Seconds)**		**Speedup**		**Parallel Efficiency**	
	YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
1	77	83	1	1	1	1
2	65	74	1.18	1.12	0.59	0.56
3	59	68	1.3	1.22	0.433	0.41
4	51s	59	1.51	1.40	0.377	0.35
5	54s	63	1.43	1.31	0.29	0.26
6	56s	65	1.375	1.27	0.23	0.21

From the tables, the difference of running time between six pseudo-clusters is not significant. Our results show that increasing the number of cores increases slightly the speed up from 1 to 4 cores, but then incurs diminishing speedup across more cores. From monitoring the system utilization during runtime, we find that this is due to network limitations of running Spark programs on one machine. As the number of partitions grows, the program becomes I/O bound, meaning that its speed is limited by the speed of input/output operations on that machine.

Table 12: Performance test for AL Module under pseudo-distributed settings.

Number of Cores α	Running Time (Seconds)		Speedup		Parallel Efficiency	
	YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
1	260s	389s	1	1	1	1
2	213s	337s	1.22	1.15	0.61	0.575
3	180s	308s	1.45	1.26	0.48	0.42
4	163s	283s	1.59	1.37	0.4	0.34
5	185s	289s	1.40	1.34	0.28	0.26
6	198s	294s	1.31	1.32	0.22	0.21

Table 13: Performance test for Recommendation Module under pseudo-distributed settings.

Number of Cores α	Running Time (Seconds)		Speedup		Parallel Efficiency	
	YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
1	456s	589	1	1	1	1
2	427s	556	1.07	1.06	0.535	0.53
3	409s	523	1.11	1.13	0.37	0.38
4	389s	509	1.17	1.15	0.29	0.29
5	395s	514	1.15	1.14	0.23	0.22
6	402s	529	1.13	1.11	0.19	0.185

Moreover, after 4 partitions, we observe that the rate of speed up decreases, since Spark has expanded the number of available cores on the machine. In fact, on a single machine, Spark sets α (the number of nodes to distribute the work across) to k, representing the total number of available cores in the machine, if α exceeds this value k (in our case we used a machine with 4 cores so $k = 4$).

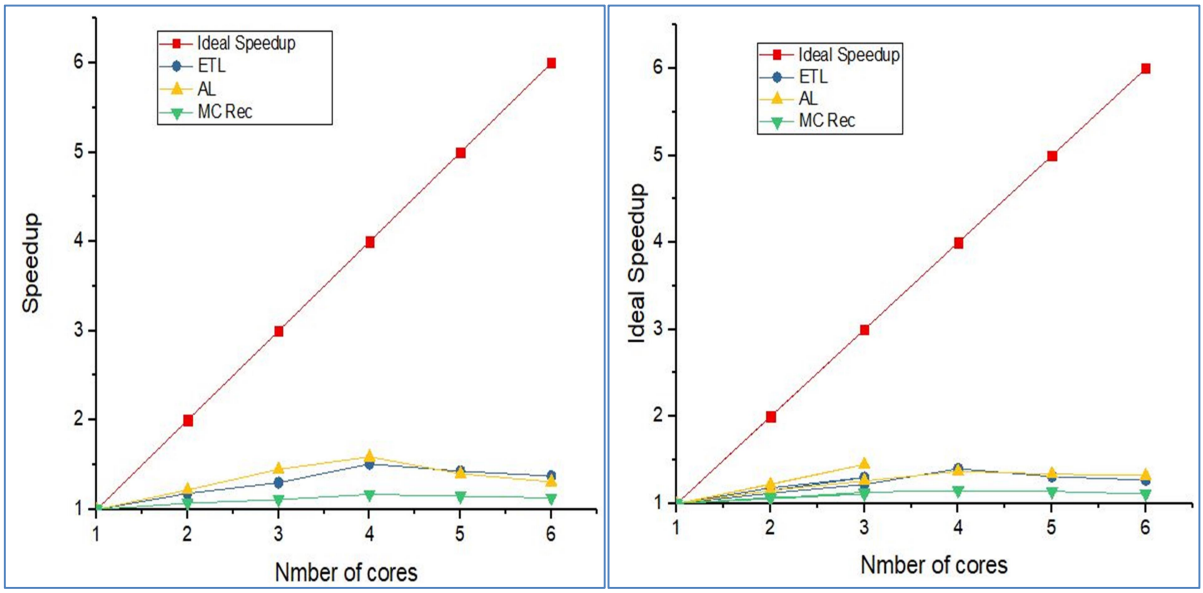


Figure 29: The Variation of the Speedup over different modules in pseudo-distributed mode: using YM (left) and HRS (right) Datasets.

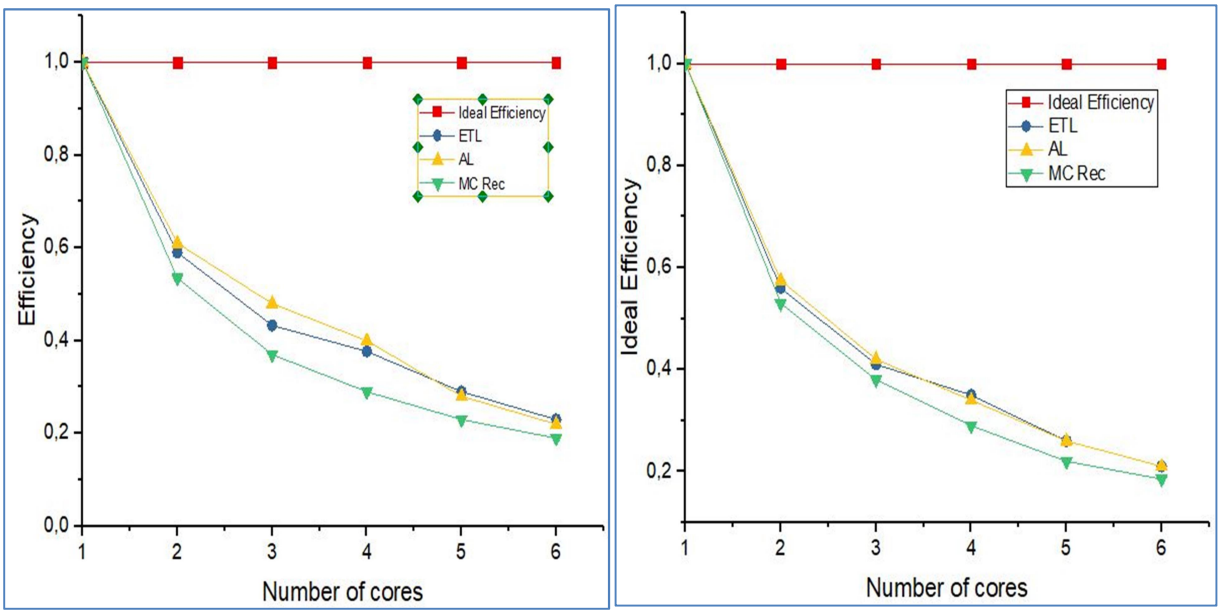


Figure 30: The Variation of the Parallel Efficiency over different modules in pseudo-distributed mode: using YM (left) and HRS (right) Datasets.

For more clarity, the obtained results are represented in Figures 29 and 30. The figure 29 shows the variation of the speedup over the three different modules by using the YM and HRS datasets. We can notice that the speedup shows little improvement when the number of the used cores is increased. In addition, the speedup can be influenced by the size of data, since it is higher in the YM dataset than that of the HRS dataset (the YM is smaller than HRS). In regards to the parallel efficiency, we notice that it decreases with the increasing number of the used cores, which demonstrates that running Spark applications in a single machine even in a pseudo-distributed mode shows some limitations and doesn't improve the efficiency of the implemented algorithms. For this reason, the next sub-section shows the results obtained in a fully-distributed mode using a cluster of many machines, which has yielded to much favorable results.

5.5.3.2. Fully-distributed

Under these settings, we used a cluster of several machines, which is composed of one master node and the rest are slaves. Hence, the number of partitions α represents the number of cluster nodes. To evaluate the scalability in a fully-distributed mode, we vary the number of partitions used α (slaves), given that the cluster with the smallest number of nodes is composed of two slaves nodes. Then, we compute the Speedup by comparing the running time through different cluster size using the equation 40. Relative size of nodes is calculated by dividing the number of nodes on each cluster by the smallest number of nodes (dividing by 2 on our case).

From the table 14, 15 and 16, the differences of running time between five clusters yielded to much more favorable results compared to the pseudo-distributed environment, since the speedup increase linearly. The summary of experimental results from tables is shown in figures 33 and 34.

Table 14: Performance test for ETL Module under fully-distributed settings.

Number of Nodes	Relative size of nodes	Running Time (Seconds)		Speedup		Parallel Efficiency	
		YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
2	1	32	40	1	1	1	1
4	2	20	29	1.6	1.37	0.8	0.68
6	3	13	17	2.46	2.35	0.82	0.78
8	4	9	11	3.55	3.63	0.88	0.90
10	5	7	8.5	4.57	4.70	0.91	0.94

Table 15: Performance test for AL Module under Fully-distributed settings.

Number of Nodes	Relative size of nodes	Running Time (Seconds)		Speedup		Parallel Efficiency	
		YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
2	1	112	179	1	1	1	1
4	2	74	113	1.51	1.58	0.755	0.79
6	3	49	74	2.28	2.45	0.76	0.81
8	4	35	47	3.2	3.8	0.8	0.95
10	5	23	37	4.86	4.83	0.97	0.96

Table 16: Performance test for Recommendation Module under Fully-distributed settings.

Number of Nodes	Relative size of nodes	Running Time (Seconds)		Speedup		Parallel Efficiency	
		YM	*HRS*	*YM*	*HRS*	*YM*	*HRS*
2	1	256	307	1	1	1	1
4	2	179	234	1.43	1.31	0.715	0.655
6	3	102	127	2.50	2.41	0.83	0.80
8	4	69	81	3.71	3.79	0.92	0.94
10	5	52	63	4.92	4.87	0.98	0.97

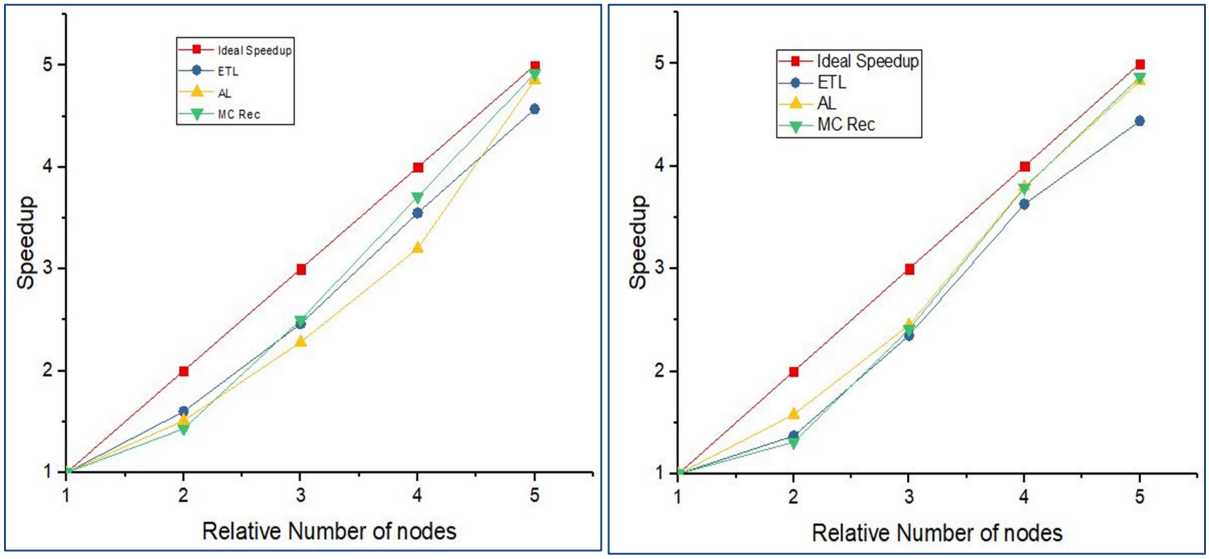


Figure 31: The Variation of the Speedup over different modules in Fully-distributed mode: using YM (left) and HRS (right) Datasets.

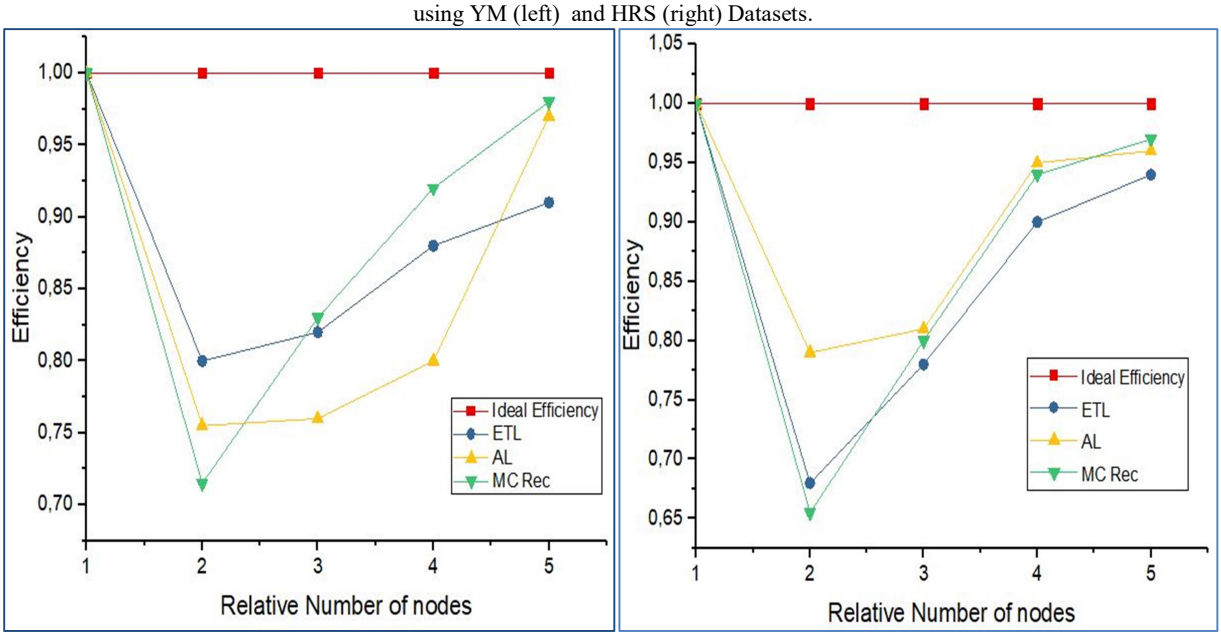


Figure 32: The Variation of the Parallel Efficiency over different modules in Fully-distributed mode: using YM (left) and HRS (right) Datasets.

We may notice that the more processing nodes are added during the processing the fastest the running time is achieved and the higher speedup is gotten. The speedup in a fully-distributed mode is linear and gets closer to the ideal speedup. Furthermore, the efficiency is higher which means that the different machines are usefully utilized to make enhance the scalability of the recommendation engine.

In fact, in order to achieve a high scalability and efficiency, careful performance optimizations have to be made in every major component in the recommendation engine [177]. To this end, we have parallelized each piece and step in the three modules composing our engine that we have detailed previously (see section 5.4.2.2.). As a consequence, the distributed algorithms were efficiently parallelized which has contributed in enhancing the speedup and efficiency. Based on the Amdahl's law, we conclude that the recommendation problem was perfectly parallel and the use of Spark to handle the parallel computing is useful.

5.5.3.3. Simulation of real-time user's profiling

The simulation of new user's profiling consists in another performance test, that we conducted to ensure the scalability of the recommendation engine and its ability to handle new users. We take into account during this experimentation, two aspects: 1) the usefulness of the constructed new users' profile and 2) the satisfaction of the new users with high-quality recommendations.

To this end, we pick at random a user that has rated over than 50 items. This user will be considered as a pseudo new-user. We mimic the behavior of a new user during the process of sign-up, by rating at each iteration 10, 20, …, 50 items. At each iteration, we construct the user profile and generate personalized recommendations based on the revealed ratings in his new profile. Then, we record the runtime needed to perform these operations. In addition, we evaluate the quality of the computed predictions. This experimentation is conducted on fully-distributed mode with 10 nodes. The obtained results are shown in the table 17.

Table 17. simulation of real-time user's profiling.

Number of collected ratings	10	20	30	40	50
Response time (s)	91	93	95	96	96
Prediction quality (MAE)	0.9647	0.9281	0.8746	0.8137	0.7921

5.6. Conclusion

In this chapter, we covered the scalability issue that we faced during our primitive experiments. This raised issue makes the use of new rising Big Data technologies a necessity to scale when the size of data tends to increase dramatically. We have illustrated how to build a scalable multi-criteria recommendation engine on apache Spark and we introduced the core concepts behind Spark's novel data flow programming model.

Using two large data sets, we provided an experimental evaluation of the implemented distributed algorithms of the proposed solutions during this thesis, using the Apache Spark under different settings (pseudo-distributed and fully-distributed). Based on the obtained experimental results the distributed algorithms using Spark have better running time than the sequential and pseudo-distributed ones. Also, When the multi-criteria rating's size is higher (HRS dataset), multi-criteria recommendation on the Apache Spark cluster has good scalability and has better running time than the sequential version. Unfortunately, the speedup of algorithms on Apache Spark pseudo-cluster was not good as expected, but the computational speedup scales linearly with a growing number of machines (fully-distributed mode).

GENERAL CONCLUSION & FUTUR PROSPECTS

"Divide each difficulty into as many parts as is feasible and necessary to resolve it."

René Descartes.

Although, the last decade was marked by a broad deployment of RSs, they suffer from some issues. One of these issues is related to the data sparsity, such as the lack of explicit ratings assigned by users to items. Indeed, the RS exploits these data in order to compute recommendations. However, if these data are insufficient, the system will be unable to identify accurate predictions. Another issue for RSs is to solve the cold start problem concerning the novelty of a user and / or item. In the case of the absence of preferences from this user and / or the preferences on this item, it becomes impossible to integrate them into the recommendations. Moreover, the accuracy of the recommendations is a major challenge for any RS since the relevance of the recommended items contributes to the satisfaction of the user's expectations and his loyalty to the service in question. Based on these research questions, we proposed in this thesis new approaches to deal with the cold-start and data sparsity issues, while ensuring high-quality recommendations (accurate predictions).

Summary of Contributions

In this section, we summarize the answers to the research questions that were raised in ***Chapter 2*** and summarize all the contributions that we have made during this thesis. Our contributions can be summarized in four main points: 1) proposing an approach called CCSDW to compute the weights of criteria and items, in order to evaluate the importance of each item compared to the others in the dataset 2) using the CCSDW method to select a set of relevant items, to bootstrap new user profile 3) proposing a hybrid recommendation algorithm based on CF and HFSM content to compensate the lack of sufficient data about new items and 4) implementing a distributed version of proposed solutions using Apache Spark, in order to gain in term of scalability and response time of algorithms. Contributions and their corresponding research questions and answers are given as follows:

1) Among the main problems confronted in recommendation context is the new user issue. Since RSs work much better for users that they know better, the more they gather information about them the more accurate predictions are made. Hence, the new user presents a big challenge that must be handled. Many questions can be raised from this issue:

- How to construct a new user profile rapidly and accurately?
- How can recommendations be generated as soon as possible, immediately after the construction of the profile for the user in question?

- To what extent the generated recommendations are satisfying for the user in question?

After conducting a detailed study of the state-of-the-art, we found that there are many techniques to resolve the new-user cold-start problem, but the most accurate one is based on the AL technique. Using this method makes the user involved during the construction of his profile, by requesting him to give his feedback about a set of well-chosen items among the whole data set. However, the selection of these items is not formalized enough and almost of times difficult to be explained to the user (i.e. the new user ignore completely why a given item is presented to him and not another). In our thesis, we answered the first research question above in ***Chapter 3*** by proposing a new method CCSDW to compute the weights of both criteria and items using multi-criteria ratings. After computing weights, the items are ranked and selected according to their importance to be shown to new users during their sign-up process. Once the ratings of the new user about the presented items are collected, the new user profile is constructed and predictions are computed to subsequently generate personalized recommendations fitting with the expressed preferences of the user in question. The proposed method is evaluated against baseline methods in term of accuracy and a user's effort to respond the third research question.

2) Another variation of the cold-start problem is the new-item issue, which consists in the absence of any feedback from users about the concerned item. In fact, handling this issue is very crucial for the continuity of the RS, since their rational is to make their users discovering new items continuously. Especially, in the case of CF approach, a new item without any ratings is impossible to be recommended since its similarity with the other items is null. Contrariwise, the CB approach doesn't suffer from this issue since it is based on the item's content to compute similarities among them and recommend similar content to users. However, the CB approach presents some shortcomings concerning the overspecialization (i.e. the generated recommendations are not diversified and don't respect the different tastes of a user), also CB doesn't ensure a high-quality of prediction (i.e. Low accuracy) when compared to CF approach. Indeed, the existing complementarity between CF and CB approaches, make them good candidates to be combined under a Hybrid RS. This verdict leads to ask many questions:

- How to combine CB and CF under one single RS to enhance each of them and cover their weaknesses?
- How to improve the accuracy of CB approach when it is used alone?

We answered the both of these questions by proposing the use of a content clustering that is rested on HFSM in ***Chapter 4***. The clustering algorithm acts on a broad textual content about the items, to classify them into groups according to their similarities. Once the clustering is conducted, the item-to-item similarities are extracted as a content similarity matrix. This resulted matrix is linearly combined with collaborative similarity matrix (constructed based on ratings), to generate the hybrid similarity matrix which is used to calculate predictions. The proposed approach was evaluated under different settings to ensure its utility in the case of new-items as well as for items belonging to the long tail. The evaluation step resulted in the outperformance of our proposition over the CF technique which is known as being better than CB.

3) Another problem that seriously affects RSs is the scalability. In fact, this kind of system is very greedy when it comes to data, i.e. the more treatments are based on a large amount of data, and the gotten results are much more precise. On the other hand, RSs suffer from performance problems when the numbers of users and products increase significantly. Furthermore, given the rushed nature of the users and the competition between the different RSs, a high-quality recommender has to deal with these challenges seamlessly. Hence, among the questions posed during our thesis project:

- How to reduce the response time of algorithms already implemented?

- How to satisfy the users of the RS by relying on two important aspects: interactivity with the system and better quality of recommendation?

In order to answer these questions and to improve scalability and response time, we have implemented a distributed version of the proposed algorithms: Active Learning (AL) and multi-criteria recommendation (MC Rec), using Apache Spark. Thanks to the use of Spark, the different functionalities are quickly ensured which facilitate the building of precise user profiles. Even changes in the preferences of the user, can be taken into account rapidly to update user's profile to continually which improve the quality of the recommendations generated.

Future Directions

In addition to the issues addressed in this thesis, which are mainly based on handling the new items and new users, leveraging multi-criteria ratings and content data and real-time features to improve the generation of recommendations in terms of high-quality, we plan to improve all the approaches presented in this thesis. Below is a list of challenging work that can further complete this thesis.

Regarding future research, there are at least five major research directions to study:

- From our point of view, this thesis constitutes a first brick for RSs based on AL. Indeed, future work aims to ask a user focused questions on the criteria that interest him instead of asking questions about the items. In this way, we can recover more information by asking only a few questions, aiming by this to minimize user's effort and maximize its satisfaction.
- In addition, we also want to address the problem of the user's context. The context is related to the interaction environment of the user with the system (professional or personal context for example). The challenge is to develop customization services offering the user at all times and on the right support, recommendations tailored to its specific context, which is likely to improve its satisfaction and retention.
- The study of the evolution of tastes over time is also part of our research perspectives. Indeed, users' assessments tend to evolve over time. Thus, our goal is to propose a system of recommendation able to detect the change of the behavior of the user and to adapt dynamically the recommendations according to the new needs of this user.
- Besides the Active Learning (AL) process, user data collected from social networks that are widely used today can be used. This will help to better know the user and learn about his trends, which will help improve the recommendation offered.
- Moreover, we plan to study also the interest of the social links for the systems of recommendation (i.e. the links resulting from the social relations such as the professional collaboration or the friendship within the framework of the platforms of the social Web) and to examine to what extent they can be complementary to the behavioral links. The aim is to evaluate the impact of this combination on the choice of neighbors and on the performance of the recommendation system in general.

AUTHOR's PUBLICATIONS

International journal articles:

- Hdioud, F., Frikh, B., Benghabrit, A., & Ouhbi, B. (2016). **Collaborative Filtering with Hybrid Clustering Integrated Method to Address New-Item Cold-Start Problem.** In Intelligent Distributed Computing IX (pp. 285-296). Springer, Cham. *Chapter Book*

- Hdioud, F., Frikh, B., & Ouhbi, B., Khalil,I. (2017). **Multi-Criteria Recommender Systems: A survey and a Method to Learn New User's Profile.** International Journal of Mobile Computing and Multimedia Communications (IJMCMC) Volume 8, Issue 4. (Accepted paper).

- Hdioud, F., Frikh, B., & Ouhbi, B., Mountasser, I. **Recommendation using a Clustering algorithm based on a Hybrid Features Selection Method.** Journal of Intelligent Information Systems (JIIS). (Accepted paper). DOI: 10.1007/s10844-017-0493-0

International conferences articles:

- Hdioud, F., Frikh, B., & Ouhbi, B. (2012, October). **A comparison study of some algorithms in Recommender Systems.** In Information Science and Technology (CIST), 2012 Colloquium in (pp. 130-135). IEEE.

- Hdioud, F., Frikh, B., & Ouhbi, B. (2013, December). **Multi-criteria recommender systems based on multi-attribute decision making.** In Proceedings of International Conference on Information Integration and Web-based Applications & Services (p. 203). ACM.

- Hdioud, F., Frikh, B., & Ouhbi, B. (2014, May). **Bootstrapping recommender systems based on a multi-criteria decision making approach.** In Next Generation Networks and Services (NGNS), 2014 Fifth International Conference on (pp. 209-215). IEEE.

- Hdioud, F., Frikh, B., Benghabrit, A., & Ouhbi, B. (2016). **Collaborative Filtering with Hybrid Clustering Integrated Method to Address New-Item Cold-Start Problem.** In Intelligent Distributed Computing IX (pp. 285-296). Springer, Cham.

REFERENCES

1. Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.
2. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval (Vol. 463). New York: ACM press.
3. Hanani, U., Shapira, B., & Shoval, P. (2001). Information filtering: Overview of issues, research and systems. User modeling and user-adapted interaction, 11(3), 203-259.
4. Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.
5. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering, 17(6), 734-749.
6. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.
7. Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender systems: Introduction and challenges. In Recommender Systems Handbook (pp. 1-34). Springer US.
8. Montaner, M., López, B., & De La Rosa, J. L. (2003). A taxonomy of recommender agents on the internet. Artificial intelligence review, 19(4), 285-330.
9. Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. In Applications of Data Mining to Electronic Commerce (pp. 115-153). Springer US.
10. Eckhardt, A. (2009). Various aspects of user preference learning and recommender systems. In DATESO (pp. 56-67).
11. Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. Expert Systems with Applications, 39(11), 10059-10072.
12. Rao, K. N. (2008). Application domain and functional classification of recommender systems--a survey. DESIDOC Journal of Library & Information Technology, 28(3), 17.
13. Burke, R. (2007). Hybrid web recommender systems. The adaptive web, 377-408.
14. Schafer, J. H. J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. The adaptive web, 291-324.
15. Massa, P., & Avesani, P. (2004, October). Trust-aware collaborative filtering for recommender systems. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 492-508). Springer Berlin Heidelberg.
16. Golbeck, J. (2006, May). Generating predictive movie recommendations from trust in social networks. In International Conference on Trust Management (pp. 93-104). Springer Berlin Heidelberg.
17. Mahmood, T., & Ricci, F. (2007). Towards Learning User-Adaptive State Models in a Conversational Recommender System. In LWA (pp. 373-378).
18. Bridge, D., Göker, M. H., McGinty, L., & Smyth, B. (2005). Case-based recommender systems. The Knowledge Engineering Review, 20(03), 315-320.
19. Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273.
20. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

21. Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-Based Systems, 46, 109-132.
22. Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.
23. Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. Machine learning, 27(3), 313-331.
24. Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.
25. Chakrabarti, S. (2002). Mining the Web: Discovering knowledge from hypertext data. Elsevier.
26. Eirinaki, M., Vazirgiannis, M., & Varlamis, I. (2003, August). SEWeP: using site semantics and a taxonomy to enhance the Web personalization process. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 99-108). ACM.
27. Gabrilovich, E., & Markovitch, S. (2006, July). Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *AAAI* (Vol. 6, pp. 1301-1306).
28. Gabrilovich, E., & Markovitch, S. (2007, January). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJcAI* (Vol. 7, pp. 1606-1611).
29. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47.
30. Billsus, D., & Pazzani, M. (1997, June). Learning probabilistic user models. In UM97 Workshop on Machine Learning for User Modeling.
31. Pazzani, M., & Billsus, D. (2007). Content-based recommendation systems. The adaptive web, 325-341.
32. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47.
33. Amatriain, X., Jaimes, A., Oliver, N., & Pujol, J. M. (2011). Data mining methods for recommender systems. In Recommender Systems Handbook (pp. 39-71). Springer US.
34. Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.
35. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994, October). GroupLens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175-186). ACM.
36. Su, X., and Khoshgoftaar, T.M. (2009). A survey of collaborative filtering techniques. In Adv. in Artif. Intell. 2009.
37. Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS), 22(1), 143-177.
38. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.
39. Koren, Y. (2010). Factors in the neighbors: Scalable and accurate collaborative filtering. TKDD, 4 (1), 2010.
40. Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In Recommender systems handbook (pp. 107-144). Springer US.
41. Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. Information retrieval, 5(4), 287-310.

42. Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: AAAI'99/IAAI '99: Proc. of the 16th National Conf. on Artificial Intelligence, pp. 439–446. American Association for Artificial Intelligence, Menlo Park, CA, USA (1999)
43. Last.fm: Music recommendation service (2009). http://www.last.fm
44. Fouss, F., Renders, J.M., Pirotte, A., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering 19(3), 355–369 (2007)
45. Karypis, G. (2001, October). Evaluation of item-based top-n recommendation algorithms. In Proceedings of the tenth international conference on Information and knowledge management (pp. 247-254). ACM
46. Bell, R and Koren, Y., (2007a). Improved Neighborhood-based Collaborative Filtering", KDD-Cup and Workshop, ACM press, 2007.
47. Bell, R., Koren, Y., and Volinsky, C. (2007b). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 95–104). New York, NY, USA. ACM.
48. Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix prize challenge. Acm Sigkdd Explorations Newsletter, 9(2), 75-79.
49. Ungar, L. H., & Foster, D. P. (1998, July). Clustering methods for collaborative filtering. In AAAI workshop on recommendation systems (Vol. 1, pp. 114-129).
50. Chee, S. H. S., Han, J., & Wang, K. (2001, September). Rectree: An efficient collaborative filtering method. In International Conference on Data Warehousing and Knowledge Discovery (pp. 141-151). Springer Berlin Heidelberg.
51. Heckerman, D. (1998). A tutorial on learning with Bayesian networks. In Learning in graphical models (pp. 301-354). Springer Netherlands.
52. Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. Machine learning, 29(2-3), 131-163.
53. Leung, C. W. K., Chan, S. C. F., & Chung, F. L. (2006). A collaborative filtering framework based on fuzzy association rules and multiple-level similarity. Knowledge and Information Systems, 10(3), 357-381.
54. Leung, C. W. K., Chan, S. C. F., & Chung, F. L. (2008). An empirical study of a cross-level association rule mining approach to cold-start recommendations. Knowledge-Based Systems, 21(7), 515-529.
55. Basu, C., Hirsh, H., & Cohen, W. (1998, July). Recommendation as classification: Using social and content-based information in recommendation. In Aaai/iaai (pp. 714-720).
56. Zanker, M., & Jessenitschnig, M. (2009, July). Collaborative feature-combination recommender exploiting explicit and implicit user feedback. In Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on (pp. 49-56). IEEE.
57. Melville, P., Mooney, R. J., & Nagarajan, R. (2002, July). Content-boosted collaborative filtering for improved recommendations. In Aaai/iaai (pp. 187-192).
58. Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2011). Developing constraint-based recommenders. In Recommender systems handbook (pp. 187-215). Springer US.

59. Felfernig, A., Friedrich, G., Jannach, D., & Zanker, M. (2015). Constraint-based recommender systems. In Recommender Systems Handbook (pp. 161-190). Springer US.
60. Aggarwal, C. C. (2016). Knowledge-based recommender systems. In Recommender Systems (pp. 167-197). Springer International Publishing.
61. Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. Artificial intelligence review, 13(5-6), 393-408.
62. Rich, E. (1979). User modeling via stereotypes. Cognitive science, 3(4), 329-354.
63. Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. AI magazine, 18(2), 37.
64. Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on (pp. 85-90). IEEE.
65. Ranganathan, A., & Campbell, R. H. (2003). An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing, 7(6), 353-364.
66. Gonzalez, G., De La Rosa, J. L., Montaner, M., & Delfin, S. (2007, April). Embedding emotional context in recommender systems. In Data Engineering Workshop, 2007 IEEE 23rd International Conference on (pp. 845-852). IEEE.
67. McNee, S. M., Riedl, J., & Konstan, J. A. (2006, April). Being accurate is not enough: how accuracy metrics have hurt recommender systems. In CHI'06 extended abstracts on Human factors in computing systems (pp. 1097-1101). ACM.
68. John, O. P., & Srivastava, S. (1999). The Big Five trait taxonomy: History, measurement, and theoretical perspectives. Handbook of personality: Theory and research, 2(1999), 102-138.
69. Tkalcic, M., & Chen, L. (2015). Personality and recommender systems. In Recommender systems handbook (pp. 715-739). Springer US.
70. Hu, R., & Pu, P. (2013, June). Exploring Relations between Personality and User Rating Behaviors. In UMAP Workshops.
71. Hu, R., & Pu, P. (2010, June). A study on user perception of personality-based recommender systems. In International Conference on User Modeling, Adaptation, and Personalization (pp. 291-302). Springer Berlin Heidelberg.
72. Quercia, D., Kosinski, M., Stillwell, D., & Crowcroft, J. (2011, October). Our twitter profiles, our selves: Predicting personality with twitter. In Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on (pp. 180-185). IEEE.
73. Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. Proceedings of the National Academy of Sciences, 110(15), 5802-5805.
74. Shen, J., Brdiczka, O., & Liu, J. (2013, June). Understanding email writers: Personality prediction from email messages. In International Conference on User Modeling, Adaptation, and Personalization (pp. 318-330). Springer Berlin Heidelberg.
75. Iacobelli, F., Gill, A. J., Nowson, S., & Oberlander, J. (2011). Large scale personality classification of bloggers. In Affective computing and intelligent interaction (pp. 568-577). Springer Berlin Heidelberg.

76. Karampiperis, P., Koukourikos, A., & Stoitsis, G. (2014). Collaborative filtering recommendation of educational content in social environments utilizing sentiment analysis techniques. In Recommender Systems for Technology Enhanced Learning (pp. 3-23). Springer New York.
77. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1-135.
78. Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014, July). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (pp. 83-92). ACM.
79. Hu, R., & Pu, P. (2010). Using personality information in collaborative filtering for new users. Recommender Systems and the Social Web, 17.
80. Tkalcic, M., Kunaver, M., Košir, A., & Tasic, J. (2011). Addressing the new user problem with a personality based user similarity measure. In First International Workshop on Decision Making and Recommendation Acceptance Issues in Recommender Systems (DEMRA 2011) (p. 106).
81. Kompan, M., & Bieliková, M. (2014, July). Social Structure and Personality Enhanced Group Recommendation. In UMAP Workshops.
82. Wu, W., Chen, L., & He, L. (2013, May). Using personality to adjust diversity in recommender systems. In Proceedings of the 24th ACM Conference on Hypertext and Social Media (pp. 225-229). ACM
83. Sharma, L., & Gera, A. (2013). A survey of recommendation system: Research challenges. International Journal of Engineering Trends and Technology (IJETT), 4(5), 1989-1992.
84. Shahabi, C., & Chen, Y. S. (2003, September). Web information personalization: Challenges and approaches. In International Workshop on Databases in Networked Information Systems (pp. 5-15). Springer Berlin Heidelberg.
85. Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002, August). Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 253-260). ACM.
86. Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender Systems: Issues, Challenges, and Research Opportunities. In Information Science and Applications (ICISA) 2016 (pp. 1179-1189). Springer Singapore.
87. Yujie, Z., & Licai, W. (2010, August). Some challenges for context-aware recommender systems. In Computer Science and Education (ICCSE), 2010 5th International Conference on (pp. 362-365). IEEE.
88. Anderson, C. (2006) The Long tail: Why the Future of Business is Selling Less of More, Hyperion Press, 2006.
89. Soboroff, I., & Nicholas, C. (1999, August). Combining content and collaboration in text filtering. In Proceedings of the IJCAI (Vol. 99, pp. 86-91).
90. Billsus, D., & Pazzani, M. J. (1998, July). Learning Collaborative Information Filters. In Icml (Vol. 98, pp. 46-54).
91. Anand, D., & Bharadwaj, K. K. (2011). Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. Expert systems with applications, 38(5), 5101-5109.
92. Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems (TOIS), 22(1), 116-142.
93. Riedl, J. (2009, October). Research challenges in recommender systems. In Tutorial sessions Recommender Systems Conference ACM RecSys.

94. Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., & Riedl, J. (2002, January). Getting to know you: learning new user preferences in recommender systems. In Proceedings of the 7th international conference on Intelligent user interfaces (pp. 127-134). ACM.
95. Anderson, C. (2006) The Long tail: Why the Future of Business is Selling Less of More, Hyperion Press, 2006.
96. Lakshmi, S. S., & Lakshmi, T. A. (2014). Recommendation Systems: Issues and challenges. IJCSIT) International Journal of Computer Science and Information Technologies, 5(4), 5771-5772.
97. Yin, H., Cui, B., Li, J., Yao, J., & Chen, C. (2012). Challenging the long tail recommendation. Proceedings of the VLDB Endowment, 5(9), 896-907.
98. Park, Y. J., & Tuzhilin, A. (2008, October). The long tail of recommender systems and how to leverage it. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 11-18). ACM.
99. Ghazanfar, M., & Prugel-Bennett, A. (2011). Fulfilling the needs of gray-sheep users in recommender systems, a clustering solution.
100. Ghazanfar, M. A., & Prügel-Bennett, A. (2014). Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems. Expert Systems with Applications, 41(7), 3261-3275.
101. Gras, B., Brun, A., & Boyer, A. (2016, July). Identifying Grey Sheep Users in Collaborative Filtering: a Distribution-Based Technique. In Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (pp. 17-26). ACM.
102. Castells, P., Vargas, S., & Wang, J. (2011). Novelty and diversity metrics for recommender systems: choice, discovery and relevance.
103. Castells, P., Hurley, N. J., & Vargas, S. (2015). Novelty and diversity in recommender systems. In Recommender Systems Handbook (pp. 881-918). Springer US.
104. Sinha, R., & Swearingen, K. (2002, April). The role of transparency in recommender systems. In CHI'02 extended abstracts on Human factors in computing systems (pp. 830-831). ACM.
105. Tintarev, N., & Masthoff, J. (2007, April). A survey of explanations in recommender systems. In Data Engineering Workshop, 2007 IEEE 23rd International Conference on (pp. 801-810). IEEE.
106. de Gemmis, M., Lops, P., Musto, C., Narducci, F., & Semeraro, G. (2015). Semantics-aware content-based recommender systems. In Recommender Systems Handbook (pp. 119-159). Springer US.
107. Campos, P. G., Díez, F., & Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Modeling and User-Adapted Interaction, 24(1-2), 67-119.
108. Voorhees, E. M. (2001, September). The philosophy of information retrieval evaluation. In Workshop of the Cross-Language Evaluation Forum for European Languages (pp. 355-370). Springer Berlin Heidelberg.
109. Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In Recommender systems handbook (pp. 257-297). Springer US.
110. Yao, Y. Y. (1995). Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science, 46(2), 133.
111. Lü, L., Medo, M., Yeung, C. H., Zhang, Y. C., Zhang, Z. K., & Zhou, T. (2012). Recommender systems. Physics Reports, 519(1), 1-49.
112. Ge, M., Delgado-Battenfeld, C., & Jannach, D. (2010, September). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In Proceedings of the fourth ACM conference on Recommender systems (pp. 257-260). ACM.

113. Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM conference on Computer supported cooperative work (pp. 241-250). ACM.
114. Bilgic, M., & Mooney, R. J. (2005, January). Explaining recommendations: Satisfaction vs. promotion. In Beyond Personalization Workshop, IUI (Vol. 5, p. 153).
115. Meyer, F. (2012). Systèmes de recommandation dans des contextes industriels (Doctoral dissertation, Université de Grenoble).
116. Pankong, N., & Prakancharoen, S. (2012). Combining algorithms for recommendation system on twitter. In Advanced Materials Research (Vol. 403, pp. 3688-3692). Trans Tech Publications.
117. Hannon, J., Bennett, M., & Smyth, B. (2010, September). Recommending twitter users to follow using content and collaborative filtering approaches. In Proceedings of the fourth ACM conference on Recommender systems (pp. 199-206). ACM.
118. Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., ... & Sampath, D. (2010, September). The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems (pp. 293-296). ACM.
119. Kislyuk, D., Liu, Y., Liu, D., Tzeng, E., & Jing, Y. (2015). Human curation and convnets: Powering item-to-item recommendations on pinterest. arXiv preprint arXiv:1511.04003.
120. Gomez-Uribe, C. A., & Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 13.
121. Wu, L., Shah, S., Choi, S., Tiwari, M., & Posse, C. (2014, October). The Browsemaps: Collaborative Filtering at LinkedIn. In RSWeb@ RecSys.
122. Golbandi N, Koren Y, Lempel R (2010, October). On bootstrapping recommender systems. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 1805-1808). ACM.
123. Golbandi N, Koren Y, Lempel R (2011, February). Adaptive bootstrapping of recommender systems using decision trees. In Proceedings of the fourth ACM international conference on Web search and data mining (pp. 595-604). ACM.
124. Rashid A M, Karypis G, Riedl J (2008). Learning preferences of new users in recommender systems: an information theoretic approach. ACM SIGKDD Explorations Newsletter, 10(2), 90-100.
125. Hdioud, F., Frikh, B., & Ouhbi, B. (2013, December). Multi-criteria recommender systems based on multi-attribute decision making. In Proceedings of International Conference on Information Integration and Web-based Applications & Services (p. 203). ACM.
126. Hdioud, F., Frikh, B., & Ouhbi, B. (2014, May). Bootstrapping recommender systems based on a multi-criteria decision making approach. In Next Generation Networks and Services (NGNS), 2014 Fifth International Conference on (pp. 209-215). IEEE.
127. Harpale A S, Yang Y (2008, July). Personalized active learning for collaborative filtering. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (pp. 91-98). ACM.
128. Elahi M, Repsys V, Ricci F (2011). Rating elicitation strategies for collaborative filtering. In E-Commerce and Web Technologies (pp. 160-171). Springer Berlin Heidelberg.

129. Elahi M, Ricci F, Rubens N (2012). Adapting to natural rating acquisition with combined active learning strategies. In Foundations of Intelligent Systems (pp. 254-263). Springer Berlin Heidelberg.
130. Elahi M, Ricci F, Rubens N (2013). Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. ACM Transactions on Intelligent Systems and Technology (TIST), 5(1), 13.
131. Elahi M, Braunhofer M, Ricci F, Tkalcic M (2013). Personality-based active learning for collaborative filtering recommender systems. In AI* IA 2013: Advances in Artificial Intelligence (pp. 360-371). Springer International Publishing.
132. Elahi, M., Ricci, F., & Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. Computer Science Review.
133. Kohrs A, Merialdo B, Improving Collaborative Filtering for New Users by Smart Object Selection. Proc. International Conference on Media Features (ICMF), 2001.
134. Sahoo N, Krishnan R, Duncan G, Callan J (2012). Research note-The halo effect in multicomponent ratings and its implications for recommender systems: The case of Yahoo! movies. Information Systems Research, 23(1), 231-246.
135. Bilge, A., & Kaleli, C. (2014, May). A multi-criteria item-based collaborative filtering framework. In Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on (pp. 18-22). IEEE.
136. Mobasher, B., Jin, X., & Zhou, Y. (2004). Semantically enhanced collaborative filtering on the web. In *Web Mining: From Web to Semantic Web* (pp. 57-76). Springer Berlin Heidelberg.
137. Dai, H., & Mobasher, B. (2007). Integrating semantic knowledge with web usage mining for personalization. *School of Computer Science, Telecommunication, and Information Systems*.
138. Jin, R., Si, L., & Zhai, C. (2006). A study of mixture models for collaborative filtering. Information Retrieval, 9(3), 357-382.
139. Popescul, A., Pennock, D. M., & Lawrence, S. (2001, August). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence (pp. 437-444). Morgan Kaufmann Publishers Inc..
140. Li, Q., & Kim, B. M. (2003, October). Clustering approach for hybrid recommender system. In Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on (pp. 33-38). IEEE.
141. Park, S. T., & Chu, W. (2009, October). Pairwise preference regression for cold-start recommendation. In Proceedings of the third ACM conference on Recommender systems (pp. 21-28). ACM.
142. Basilico, J., & Hofmann, T. (2004, July). A joint framework for collaborative and content filtering. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 550-551). ACM.
143. Hdioud, F., Frikh, B., Benghabrit, A., & Ouhbi, B. (2016). Collaborative Filtering with Hybrid Clustering Integrated Method to Address New-Item Cold-Start Problem. In Intelligent Distributed Computing IX (pp. 285-296). Springer, Cham.
144. Sebastiani, F. (2002). Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), 1-47.

145. Willett, P. (2006). The Porter stemming algorithm: then and now. Program, 40(3), 219-223.
146. Benghabrit, A., Frikh, B., Ouhbi, B., Zemmouri, E. M., & Behja, H. (2013, December). Text Document Clustering with Hybrid Feature Selection. In Proceedings of International Conference on Information Integration and Web-based Applications & Services (p. 600). ACM.
147. Li, Y., Luo, C., & Chung, S. M. (2008). Text clustering with feature selection by using statistical data. IEEE Transactions on Knowledge and Data Engineering, 20(5), 641-652.
148. Li, Q., & Kim, B. M. (2003, October). Clustering approach for hybrid recommender system. In Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on (pp. 33-38). IEEE.
149. Puntheeranurak, S., & Tsuji, H. (2007, October). A multi-clustering hybrid recommender system. In Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on (pp. 223-228). IEEE.
150. Park, Y. J., & Tuzhilin, A. (2008, October). The long tail of recommender systems and how to leverage it. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 11-18). ACM.
151. Wen, J., & Zhou, W. (2012). An improved item-based collaborative filtering algorithm based on clustering method. Journal of Computational Information Systems, 8(2), 571-578.
152. Sun, D., Li, C., & Luo, Z. (2011, August). A content-enhanced approach for cold-start problem in collaborative filtering. In Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on (pp. 4501-4504). IEEE.
153. Strehl, A., Ghosh, J., & Mooney, R. (2000, July). Impact of similarity measures on web-page clustering. In Workshop on artificial intelligence for web search (AAAI 2000) (Vol. 58, p. 64).
154. Benghabrit, A., Ouhbi, B., Behja, H., & Frikh, B. (2013). Statistical and Semantic Feature Selection for Text Clustering. Journal of Intelligent Computing Volume, 4(2), 69.
155. Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. Recommender systems handbook, 257-297.
156. Cremonesi, P., & Turrin, R. (2009, October). Analysis of cold-start recommendations in IPTV systems. In Proceedings of the third ACM conference on Recommender systems (pp. 233-236). ACM.
157. Adomavicius, G., Manouselis, N., & Kwon, Y. (2011). Multi-Criteria Recommender Systems. Recommender systems handbook, 1, 769-803.
158. Wijayanto, A., & Winarko, E. (2016, October). Implementation of multi-criteria collaborative filtering on cluster using Apache Spark. In Science and Technology-Computer (ICST), International Conference on (pp. 177-181). IEEE.
159. Owen, S., & Owen, S. (2012). Mahout in action.
160. Schönberger V. M., Cukier K., "Big Data: A revolution that will transform how we live, work, and think". New York: Houghton Mifflin Harcourt, 2013.
161. Sagiroglu, S., & Sinanc, D. Big data: A review. In : Collaboration Technology
162. Jamiy, F. E., Daif, A., Azouazi, M., & Marzak, A. (2015). The potential and challenges of Big data-Recommendation systems next level application. arXiv preprint arXiv:1501.03424.
163. Amatriain, X. "The recommender problem revisited," In : Proceedings of the 8th ACM Conference on Recommender systems. ACM, 2014. p. 397-398.

164. Walunj, S. G., & Sadafale, K. “An online recommendation system for e-commerce based on apache mahout framework,” In : Proceedings of the 2013 annual conference on Computers and people research. ACM, 2013. p. 153-158.
165. Seminario, C. E., & Wilson, D. C. (2012, September). Case Study Evaluation of Mahout as a Recommender Platform. In RUE@ RecSys (pp. 45-50).
166. Casey, W. E. (2014). Scalable collaborative filtering recommendation algorithms on apache spark.
167. Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. Lecture Notes in Computer Science, 5034, 337-348.
168. Verma, J. P., Patel, B., & Patel, A. "Big data analysis: recommendation system with Hadoop framework," In : Computational Intelligence & Communication Technology (CICT), 2015 IEEE International Conference on. IEEE, 2015. p. 92-97.
169. kumar Bokde, D., Girase, S., & Mukhopadhyay, D. (2015). An item-based collaborative filtering using dimensionality reduction techniques on Mahout framework. In 4th Post Graduate Conference for Information Technology (iPGCon-2015), Sangamner, Published in Spvryan’s International Journal of Engineering Science and Technology (SEST) (pp. 2394-0905).
170. Zhao, Z. D., & Shang, M. S. (2010, January). User-based collaborative-filtering recommendation algorithms on hadoop. In Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on (pp. 478-481). IEEE.
171. Jiang, J., Lu, J., Zhang, G., & Long, G. (2011, July). Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop. In Services (SERVICES), 2011 IEEE World Congress on (pp. 490-497). IEEE.
172. Schelter, S., Boden, C., & Markl, V. (2012, September). Scalable similarity-based neighborhood methods with MapReduce. In Proceedings of the sixth ACM conference on Recommender systems (pp. 163-170). ACM.
173. Kupisz, B., & Unold, O. (2015, March). Collaborative filtering recommendation algorithm based on Hadoop and Spark. In Industrial Technology (ICIT), 2015 IEEE International Conference on (pp. 1510-1514). IEEE.
174. Almohsen, K. A., & Al-Jobori, H. "Recommender Systems in Light of Big Data," International Journal of Electrical and Computer Engineering, 2015, vol. 5, no 6.
175. Panigrahi, S., Lenka, R. K., & Stitipragyan, A. "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark," Procedia Computer Science, 2016, vol. 83, p. 1000-1006.
176. Chowdhury, M., Zaharia, M., & Stoica, I. (2014). Performance and scalability of broadcast in Spark. 2014-10-08]. http://www. cs. berkeley. edu/~ agearh/cs267. sp10/files/mosharaf-spark-bc-report-spring10. pdf.
177. Cambazoglu, B. B., & Baeza-Yates, R. (2016, July). Scalability and efficiency challenges in large-scale web search engines. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (pp. 1223-1226). ACM.
178. Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. Recommender systems handbook, 107-144.

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

Résumé de la thèse

De nos jours, avec l'émergence du web, la prise en compte de l'utilisateur lors du processus de recherche d'information est devenue une nécessité. Ce défi concerne la satisfaction des attentes des utilisateurs et leur rétention ; qui est connu sous le nom de personnalisation. En effet, la personnalisation connue sous le nom de Systèmes de Recommandation est un domaine de recherche qui attire l'attention de nombreux chercheurs dans le but de proposer les produits les plus proches des goûts de chaque utilisateur.

Cette thèse couvre l'étude des systèmes de recommandation, en mettant l'accent sur les différents types de problèmes de démarrage à froid.

Dans une première contribution, nous avons résolu le problème du démarrage à froid des nouveaux utilisateurs. Pour cela, nous avons proposé un processus d'apprentissage actif basé sur une méthode de pondération des articles que nous avons appelé CCSDW. Cette méthode repose sur l'exploitation des évaluations multicritères pour évaluer l'importance des critères et calculer leur poids. Ensuite, les poids des critères sont utilisés pour calculer les poids des produits et les réordonner pour qu'ils soient évalués ensuite par le nouvel utilisateur au cours du processus de l'Active Learning (AL).

Un autre problème important sur lequel nous nous sommes focalisés est celui du nouveau produit. Nous avons résolu ce problème en utilisant un nouvel algorithme de clustering du contenu basé sur la méthode HFSM (Hybrid Features Selection Method). Cette méthode permet de calculer les similarités entre les produits en fonction de leur contenu ce qui permet de compenser les votes manquants soit dans le cas d'items appartenant à la longue queue, ou bien pour les nouveaux items.

Notre dernière contribution vise à intégrer toutes les approches développées précédemment dans une infrastructure de recommandation distribuée en exploitant les outils Big Data. Pour cela, nous avons établi un moteur de recommandation à grande échelle par l'outil Spark. Les résultats expérimentaux ont montré une grande amélioration en terme du temps de calcul et en terme de qualité de recommandation.

Mots clés : Recommandations, personnalisation du contenu web, profilage des utilisateurs, démarrage à froid, recommandation à temps réel, recommandation à grand échelle