وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR- ANNABA

BADJI MOKHTAR UNIVERSITY-ANNABA

جامعـــة باجـــي مختـــار عنـابـة-عنابة

Année : 2021

Faculté des Sciences de L'Ingéniorat

Département d'informatique

THESE

Pour obtenir le diplôme de docteur 3éme cycle

Etude de la Diversité, la Nouveauté, et la Pertinence dans les Systèmes de Recommandation

Filière: Informatique

Spécialité: Sciences et Technologies de l'Information et de la Communication.

Préparé par

Chems Eddine Berbague

JURY :

Président (e):	Khadir Mohamed Tarek	Pr.Université Badji Mokhtar, Annaba
Directrice de thèse :	Hassina Seridi	Pr.Université Badji Mokhtar, Annaba
Co-dicteur de thèse :	Karabadji Nour El-Islam	Dr.Université Badji Mokhtar, Annaba
Examinateur :	Zarzour Hafed	Dr. Mohamed-Cherif Messaadia, Souk Ahras
Examinateur :	Suici-Meslati Labiba	Pr.Université Badji Mokhtar, Annaba
Examinateur :	Kouahla Zine Eddine	Dr. Université 08 Mai 1945, Guelma
Invité :	Panagiotis Symeonidis	Pr. Université de Aegean, Gréce

This thesis may be of interest to:

Student, and researchers who work on e-commerce, movie recommendation,recommender systems, collaborative filtering,collective intelligence, optimization, and/or evolutionary algorithms.

Readers from all other backgrounds interested in discovery

Foreword

Acknowledgment

I would express my deep feeling of gratitude to everyone who contributed in the preparation of this thesis.

I am thankfull to everyone who supported me during my five years of research, and to those who gave me advices, and shared their experiences, knowledge, wisdom, or just wished me success.

I dedicated my sincere thanks to my parents, grandparents, brothers, sisters, friends, and colleagues from my university Badji Mokhtar-Annaba, or other universities.

I appreciate the kind help of my professors in LABGED laboratory, and in particular my supervisor HASSINA SERIDI, and my co-supervisor Nour El-Islem Karabadji.

I am grateful to computer science department of the Free University of Bolzano, for their kind hospitality during my internship, and I thank professor PANAGIOTIS SYMIONIDIS for his help.

Lastly, I dedicate this thesis to everyone interested by discovery.

ملخص

إن أنظمة التوصية حقل بحث نشط خصوصا في مجال التجارة الإلكترونية يهدف إلى إرضاء المستعملين. مهمة نظام التوصية هي نمذجة سلوك المستعملين بالتنقيب في معطياتهم السابقة والتعرف إلى أنماط تفضيلاتهم ثم توصية عناصر مهمة. عرفت مثل هذه الأنظمة إستعمالا في التجارة الالكترونية، تطبيقات الوسائط الاجتماعية، وشخصنة الويب.

خوارزمية التصفية التعاونية المبنية على الذاكرة واحدة من تقنيات البحث الأكثر جذبا والمعروفة بفاعليتها، بساطتها، وسهولة إقامتها. هذه الخوارزمية تختار لكل مستعمل مستهدف جوارا من الأعضاء المتماثلين لإستعمالهم من أجل توصية عناصر مفضلة.

غير أن فاعلية هذه الخوارزميات ما تزال محدودة من حيث تنوع التوصيات ومشكل قابلية التوسع لأن قدرات متزايدة من الموارد ضرورية لضمان وظيفية النظام الحسنة.

في هذه الأطروحة حاولنا حل مشكل التنوع بالأخذ في الحسبان تماثل وتنوع كل زوج من المستعملين واستغلال هذه المعلومة لإقتراح بدائل أفضل من خوارزميات التجميع التقليدية. جوهر مقترحنا يرتكز على إستعمال الخوارزميات الجينية لحل مشكل تجميع ذي دالة ملاءمة متعددة الأهداف و بإعتبار تنوع و دقة التوصيات.

في مقترحاتنا, بحثنا عن دمج كل الإعدادات ضمن تمثيل جيني واحد ما سمح لنا بالحصول في نفس الوقت على مجموعات متماثلة ومتنوعة من المستعملين. نوعية المجموعات النهائية سمحت لنا بالتحكم في مستوى التنوع الذي نعينه لكل مستعمل.

كلمات البحث: تجميع، خوارزمية جينية ،حداثة، التنوع، تصفية تعاونية.

Abstract

Recommender systems are an active research filed essentially in the domain of e-commerce, whereas the objective is to satisfy the users. The main task of a recommender system consists of modeling the users' behaviour by mining their past data to identify their preferences patterns, and recommending them the most interesting items. Such systems have known use in e-commerce, social media applications, and web personalizing.

Memory-based collaborative filtering algorithm is among the most attractive recommendation techniques, known by its efficiency, simplicity and ease deployment. This algorithm selects for every target user a neighborhood of same-minded members to use them for recommending relevant items.

Yet, the efficacy of these algorithms is still limited in terms of the recommendation diversity and the scalability problem whereas increasing resources capacities are required to ensure the proper functioning of the system.

In this thesis, we address the diversity problem by considering both the similarity and the diversity of each pair of users to propose other alternatives than the conventional clustering algorithms. Our main proposition consists of using genetic algorithms to optimize a clustering problem with a multi-objectives fitness function that considers both diversity and relevancy of the recommendations.

In our propositions, we try to integrate all the required settings in one genetic representation. This method allowed us to get groups of similar and diverse users at once. The quality of the final clusters allowed us to control the diversity level which we assign to each user.

Keywords: clustering, genetic algorithm, novelty, diversity, collaborative filtering.

Résumé

Les systèmes de recommandations sont un champ de recherche actif esentiellement dans le domaine du e-commerce dans le but de satisfaire les utilisateurs. Dans les systèmes de recommandation, la tâche principal consiste à modéliser le comportement des utilisateurs par la fouille de leurs données passées pour identifier leurs préférences et leur recommander les éléments les plus intéressants. Ces systèmes sont vulgarisés de nos jours et utilisés dans plusieurs domaines tel que l'e-commerce, les réseaux sociaux, et la personnalisation du web.

L'algorithme de filtrage collaboratif basé mémoire est parmi les techniques de recommandation les plus utilisés, connu par leur efficacité, simplicité, et faciliter de déploiement. Cet algorithme sélectionne pour chaque utilisateur cible un voisinage des membres similaires pour recommander des éléments pertinents.

Cependant, l'efficacité de ces algorithmes reste encore limitée en termes de diversité des recommandations et de scalabilité où les ressources croissantes sont requises pour assurer le bon fonctionnement du système.

Dans cette thèse, nous abordons le problème de la diversité en considérant la similarité et la diversité de chaque pair d'utilisateurs afin de proposer des alternatives aux algorithmes de clustering conventionnels. L'essence de notre proposition consiste à utiliser des algorithmes génétiques pour optimiser des éléments de recommandation lors du clustering, en utilisant une fonction de fitness multi objectives et en considèrant la diversité et la pertinence des recommandations.

Dans nos propositions, nous cherchons à intégrer tous les paramètres requis dans une seule représentation génétique, ce qui nous a fournis des groupes d'utilisateurs similaires et divers en même temps. La qualité des groupes finaux nous a permis de contrôler le niveau de diversité affecté à chaque utilisateur.

Mots-clés: *clustering, algorithme génétique, nouveauté, diversité, filtrage collaboratif.*

Publications

The content of the thesis is the effort of 4 years research work. The experimentions and results presented in Chapter 4 and Chapter 5 are taken and extended from our research publications.

International journals

- Chems Eddine Berbague, Nour El-islem Karabadji, Hassina Seridi, Panagiotis Symeonidis, Yannis Manolopoulos, Wajdi Dhifli, An overlapping clustering approach for precision, diversity and novelty-aware recommendations, Expert Systems with Applications, Volume 177, 2021, 114917, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.114917.
- Chems Eddine Berbague, Nour El-islem Karabadji, Hassina Seridi, Panagiotis Symeonidis, Markus Zanker, An Evolutionary-based Approach for Providing Accurate and Novel Recommendations: In Int. J. of Business Intelligence and Data Mining.

International conferences

- Berbague Chemseddine, Karabadji Nour El-Islem, Seridi Hassina. (2019) Enhancing the Sales Diversity Using a Two-Stage Improved KNN Algorithm. In: Chikhi S., Amine A., Chaoui A., Saidouni D. (eds) Modelling and Implementation of Complex Systems. MISC 2018. Lecture Notes in Networks and Systems, vol 64. Springer, Cham, https://doi.org/10.1007/978-3-030-05481-6_15.
- Berbague Chemseddine, Karabadji Nour El-Islem, Seridi Hassina. (2018) An Evolutionary Scheme for Improving Recommender System Using Clustering. In: Amine A., Mouhoub M., Ait Mohamed O., Djebbar B. (eds) Computational Intelligence and Its Applications. CIIA 2018. IFIP Advances in Information and Communication Technology, vol 522. Springer, Cham. https://doi.org/10.1007/978-3-319-89743-1_26

List of acronyms

We use in this thesis the acronyms summarized in the next table for the sake of readability. The meaning of an acronym or an abbreviation is only cited in its first use.

Acro	Explanation
RS	Recommender system
UCF	User-based collaborative filtering
ICF	Item-based collaborative filtering
CBF	Content-based filtering algorithm
MF	Matrix factorization
DEM	Demographic-based filtering algorithm
SOC	Social-based filtering algorithm
GA	Genetic algorithm
EA	Evolutionary algorithm
MAE	Mean average error
RMSE	Root mean square error
DL	Deep learning
MeCF	Memory-based collaborative filtering
MoCF	Model-based collaborative filtering
LT	Long tail problem

Contents

List of Figures

List of Tables

When I consider what people generally want in calculating, I found that it always is a number. I also observed that every number is composed of units, and that any number may be divided into units.

Muhammad ibn Musa al-Khwarizmi

1 Introduction

▷ *A recommender system (RS) is a great tool that allows effective access to resources online. RS covers a wide range of recommendation applications such as books, movies, music, jokes, and more social networks. RS has appeared as a consequence of adapting existing tools and technologies taken from the information retrieval field. RS aims to satisfy the users by offering relevant and diverse recommendations. In this chapter, we present an introduction to the subject under study, in which we show the importance of our research and the details of our contributions.* ◁

Chapter outline

1.1 General context

REcommender systems are the application of information retrieval theory in e-commerce websites, online services, and social websites...etc. In this domain, RS aims to match a set of users (i.e., or clients.) to items (i.e., or products.) which they may like. To this end, a collection of information includes users information (e.g., demographic information, social information, and behavioral information.), items information (e.g., meta-data information, tags, and descriptions.), and contextual information (e.g., time and location) were underuse. A RS efficacy may be limited because of two major drawbacks:

The recommendation quality: Although conventional recommender systems only targeted the relevancy of recommendations, new recommendation algorithms tend to give more importance to diversity as a critical criterion to satisfy the needs of the users. In effect, balancing relevancy and diversity is a hard task that involves adequate techniques because of the contradiction of these metrics.

The scalability of the system: Real-world applications grow rapidly and the need for computation and storage resources increases, whereas new users and products are added daily. Material based-solutions and machine learning-based solutions such as clustering were proposed to alleviate this problem. However, computational optimization techniques are a substantial solution to conserve time and energy.

This thesis presents recommender systems' state-of-the-art, followed by the proposed solutions and progressive analysis of this research domain through the collaborative filtering algorithms which is a known recommendation approach.

1.2 Thesis context

RS field consists of many algorithms which depend on many factors such as the desired recommendation quality, and the available information.

Our interest is to develop a memory user-based collaborative filtering algorithm that is one of the known, simplest, and efficient recommenders. In particular, we focus on improving its performances in terms of a) the recommendation quality (i.e., balancing diversity and relevancy.), and b) the scalability problem (i.e., alleviating the computation complexity.). Our steps to achieve these objectives can be summarized in the next items:

- Analyzing the diversification process using baseline recommenders, then exploring the use of genetic algorithms as a key technique to improve the recommendation quality.
- Exploring the use of clustering techniques to improve the scalability and the recommendation diversity.

1.3 Motivations

Recommendation process efficiency depends on many axes which are resources and quality requirements. From one side, the rapid growth of the web accompanied by the increasing users' number implies more work on improving the scalability. From the other side, the recommendation quality definition is changing, whereas a contradictory multi-criteria quality is targeted.

In memory user-based collaborative filtering, the number of computations depends on the number of users, whereas for N user $N \cdot N - 1$ similarity value is computed. These values are, then, used to select for each user the closest neighbors. In general, the neighborhood selection affects the quality of the recommendations, whereas, data sparsity decreases the efficiency of selecting beneficial neighbors. In fact, the users' similarity is calculated by considering the co-rated items which may be very limited, hence the obtained similarity may be unreliable. In fact, the clustering permits to re-allocate the set of users or items into groups of similar profiles. This approach increases the intra-class similarity and decreases the inter-class similarity which enhances the performance of the neighborhood selection, and the recommendation quality consequently. However, the available clustering algorithms are mostly stochastic approaches and are applied to sparse statistical features of individuals. In this area, meta-heuristics including genetic algorithms have achieved considerable advancements to deal with these problems.

Additionally, novelty, diversity, and accuracy represent the keystones of the new recommendation fashion. Currently, users seek to get more than relevancy. A pleasant recommendation experience would hold some novelty and diversity within the recommendation set. In fact, recommender systems were used for both desktop and mobile applications. The recommendation ranking has become an interesting matter because of the hard content exploration issue and the limited positions of small devices' screens. Taking an example of two movies that are equally preferred by a typical user who may give a rating of 5 stars. A question is raised about which movie to put in the first position. On the basis of this example, recent works tended to adopt a third evaluation category said rank evaluators, whereas the proposed metrics evaluate novel and diverse items with higher scores. Thus, the answer to the question above would be selecting the movie which surprises the user. Also, to ensure a higher chance to sell a product, the recommendations set should contain diverse items. A greater choice, in the case of the movies market, would correspond to a recommendation set of different movies genres, so a user who likes drama and action would get recommendations from both of these categories.

1.4 Objectives

Through the research done during the preparation of this thesis:

Firstly, we reviewed the state-of-the-art of recommender systems and analyzed their known limitations and issues, then summarized the definitions, classifications, and comparisons of the different proposed contributions. In particular, we gave more attention to its most pressing issue which is the recommendation quality. To this end, we used an evolutionary algorithm, which is a new trend in the field, known for its effectiveness in such kind of combinatorial problems.

Then, we tried to improve the recommendation quality in parallel with the scalability problem. Thus, we seek to adopt a clustering formulation that targets both issues at once.

1.5 Contributions

In this thesis, we have evaluated the memory user-based collaborative filtering in term of the quality of recommendations.

Firstly, we tested the effect of adjusting the statistical features of users to improve the performance of the recommendation algorithm. In this stage, we have employed known metrics from the state-of-the-art and evaluated their effect on the recommendation quality.

Then, we passed to evaluate the collaborative filtering using a genetic algorithm, whereas we criticized the stochastic-based seeds initialization in K-means and proposed a better clustering scheme in which we benefited from a clearer quality indicator during the evolutionary process.

The encouraging results of the genetic algorithm pushed us to continue investigating the benefits of clustering to increase the recommendation diversity. In effect, we proposed a genetic-based optimization algorithm that assigns the users binary weights (i.e., 1 or 0.), to improve the recommendations coverage. Then, we extended this proposition to a better clustering algorithm, which tackles the diversification problem and the scalability problem.

As another effort to explore the diversification task in collaborative filtering algorithms, we analyzed the effect of selecting a beneficial neighborhood to improve novelty. To this end, we proposed a fitness function that increases the probability of recommending novel and relevant items.

The experimental work done under the research of this thesis was performed on two well-known benchmarks from Movielens.

1.6 Thesis structure

In **Chapter 1**, we presented a general introduction to the thesis. In the rest of the chapters, we present the state of the-art of recommendation systems and more details about our contributions as in the next structure:

Chapter 2: A general presentation of the recommender system field including definitions, recommendation algorithms, recommendation evaluations, and challenges.

Chapter 3: A detailed presentation of memory-based collaborative filtering as the main algorithm which we developed, whereas we cite different diversification techniques, comparisons, and technical related requirements.

Chapter 4: We present the use of genetic algorithms to improve the recommendation relevancy. In our contributions, we criticized the selection of the initial seed in the *k*-means algorithm and presented the use of genetic-based clustering algorithm. In more detail, we tried on the level of each cluster to minimize a prediction error to improve precision. Furthermore, we developed our algorithm to improve the recommendation novelty/diversity and the scalability using a new clustering formulation.

Chapter 5: We improved memory user-based collaborative filtering using a genetic algorithm. In effect, we analyzed the profile of each user to amount the recommendations novelty which a possible neighbor may offer to the target user. Then, we optimized the quality of neighborhood members using a genetic algorithm by increasing the probability of recommending novel and more diverse items.

Chapter 6: In this chapter, we present general conclusions, in which we discussed the work done during the preparation of this thesis and possible extensions in future works.

Science is what we understand well enough to explain to a computer. Art is everything else we do.

Donald Knuth

2

General introduction to recommender systems

▷ *In this chapter, we define recommender systems and their recommendation algorithms, evaluation metrics, applications, limitations, and comparison to concepts from the same context. Later, we come to list the recent known trends of this domain.* ◁

Chapter outline

2.1 Introduction

THe end of the 1980s, the web was created as a tool of communication and sharing resources online and have been developed successively [Berners-Lee 01]. Nowadays, millions of users reach the web every day to look for the content of their preferences. In such a field, search engines (SE) [Croft 10] play a critical role in facilitating and accelerating the exploration of the vast wide of offered content and overlapped information. Similarly, information management systems such as public libraries, numerical archives, and online stores use the same techniques to locate resources that fit users' needs. SEs raise two issues: understanding users' needs and matching their queries to databases content.

Although databases seem to have a clear and specific function which is answering systematic queries by automatically applying internal correspondence tasks, web search adopts techniques such as text mining, media content analysis, statistical approaches to isolate the decisional concepts, which allow defining the users' needs. As an example, TF-IDF is a weighting technique that can be performed on the users' data. It gives, for each term, an importance weight by analyzing its appearance frequency in a corpus of text. This technique allows representing a document as a vector of less and more significant terms, which facilitates the correspondence matching between the web resources and the users' queries. The process of managing resources on the web belongs to web mining techniques [Kosala 00], which target the web structure, the web content, and the users' models. In the end, each user receives pertinent results which he asked for using a natural language or keywords.

In the case of web search, the users know exactly what they are looking for. However, most search systems face issues at eliminating the confusion related to understanding their queries. For example, in the case of a user who is looking for Java language lessons, a problem is raised because of the confusion between the possible concepts: Java programming language, Java island, or Java coffee. This problem is usually faced by considering additional information such as location, time, and search history. Simple information telling that the user is a programmer may eliminate such confusion.

From a different perspective, in online services such as e-learning, e-commerce, and tourism applications, the users do not know what they are in need to, they just explore the available content in hope of getting/observing an interesting product that may excite them, hence, buying/consuming it. A great example of an e-commerce application is movies markets, whereas the Netflix website is considered as a reference in the world with total revenue of 8.83 billion U.S. dollars achieved in 2016. This website offers streaming access to a huge media library for around 100 million users recorded in the second quarter of 2017. Each user explores the market with the aim of getting content that may please him and expects the system to recommend him relevant products. To this end, in 2009 Netflix has opened a competition to win 1 million dollars for the team who can enhance the accuracy of their system by 10 % [Bennett 07].

Recommender systems (RSs) [Isinkaye 15] were developed in parallel with web mining techniques, whereas, in an earlier age, recommender systems have adopted many techniques of the information retrieval field such as text mining tools. The objective of a recommender system is to benefit from the available information to enhance the users' experiences and to increase the revenues of the system at once. Nowadays, richer content is available online and covers a considerable scope of types such as the users' demographic information, the products' meta-data, behavioral information, social information, and contextual information. In the next sections, we discuss in more detail the origins of recommender systems.

2.2 Recommender systems

RS is the fruit of exploiting many other research fields such as cognitive science, approximation theory, information retrieval, forecasting theories, management science, and the marketing [Adomavicius 05a]. RS field knew a successive and fast development due to its impact on the users' experiences. This development is pushed by the next motivations [Ricci 15]:

- RS is developed in every domain including the biggest social media websites such as LinkedIn, Facebook, and media websites such as Spotify, YouTube, Last.fm, Netflix, and IMDb in addition to e-commerce websites such as Amazon.com.

- Academic contributions are on focus due to the foundation of many conferences and workshops dedicated specifically to the field such as Recommender Systems (RecSys), in addition to conferences in the same scope such as domains of information retrieval, user modeling, and intelligent user interfaces.

- RS knows the importance in the level of educational institutions, in computer science conferences, and publishers such as Springer, whereas several books in RS were published. Moreover, many soft-wares were developed for RS.

- Several special issues in academic journals have covered the RS field. Among these journals: AI Communications (2008), IEEE Intelligent Systems (2007), International Journal of Electronic Commerce (2006), International Journal of Computer Science and Applications (2006), User Modeling and User-Adapted Interaction (2014, 2012), and ACM Transactions on Intelligent Systems and Technology (2015).

2.2.1 Notations

In a recommender system, we have a set of users denoted by U. These users aim to exploit a catalog of items I to select interesting products. An item $i \in I$ can be a movie, a piece of music, a book, or else. A typical user u may express implicitly or explicitly a reaction toward a given item i. If we consider the actions "listening to music" or "watching a movie", we can define a binary evaluation matrix R whose values belong to

$U \times I$, whereas each value is 1 or 0. Elements of R are denoted by $R_{u,i}$. In more explicit evaluations, users can express a rating value in the range {0,1,2,..,max_{rate}}, where Max_R is the maximum rating value and could be 5, 10...etc. The value 0 means the absence of an evaluation, while the values from 1 to Max_R reflect the admiration of an item by a user. For an item j not seen by a user u, we look to predict the rating $\bar{R_{u,j}}$ to compose a recommendations set R_u^I. The length of R_u^I depends on the deployed application. More precisely, mobile applications involve a shorter recommendation set than desktop applications. We denote by N the size of R_u^I. Elements of R_u^I are ranked according to a croissant order of the highest predicted values. The relevancy of j to u is evaluated according to a minimum threshold θ, in other terms if $j \in R_u^I$ and $R_j^u > \theta$ then j is relevant to u. Moreover, items seen, purchased, or evaluated by user u are denoted by I_u and we note it by u's profile. Similarly, users, who have seen, purchased, or evaluated an item i are denoted by U_i and we note it by i's profile.

2.2.2 Definition

We can define a recommender system formally by a function that seeks to link users to their most preferred items using data mining tools. Let's consider a set of data sources as in the form $D = \{Dem, Cont, Cntx, Beh\}$, which denotes demographic information, content information, contextual information, and behavioral information respectively. Furthermore, we consider a feature extraction function F_{ex} which takes as an entry one or more items of D. Also, we consider a prediction function F_{pred} with takes as parameters the set of users U, the set of items I, and a set of available data about both users and items D. F_{pred} can be an aggregation formula or a machine learning tool that predicts the users' preferences $\hat{R}$ and we write:

$$\hat{R} \leftarrow F_{pred}(F_{ex}(D), U, I) \tag{2.1}$$

2.2.3 Motivations

RS appeared as a consequence of the information overload which requires important techniques to satisfy the users. In fact, there are various reasons which justify the need for an online system to exploit this technology [Kaminskas 17] [Ricci 15]:

Increase the recommendation coverage: RS aims to cover more items while recommending to users, which serves the sellers as well [Yin 12].

Sell more diverse items There is a high number of items undersell in e-commerce markets, which confuses the users and makes exploring and selecting items a hard task. Thus, a RS should offer relevant and diverse recommendations at once [Bradley 01].

Increase user satisfaction. Satisfying a user may be accomplished by offering relevant, surprising, and diverse recommendations. High recommendation quality will increase the reliability of the recommendation system. Thus, it improves the revenues.

Increase the user attraction. An RS should model more perfectly loyal users by exploiting their data. As the amount of the entered data increases, the recommendation quality should become higher. A loyal user may increase the possibility to attract more new users to the system.

Better modeling the user preferences Another RS task is to understand what a given user likes. Experiences showed that using more data about the users, the items, and the context allows to increase the recommendation relevancy.

2.2.4 Recommender system design

RS exploits a variety of information about the users, the items, the users' preferences, the relationships between users, the items' meta-data, and the context of the recommendation. This information is used to identify, for each user, a set of relevant recommendations. According to [Bobadilla 13] recommendation generation involves the next requirements:

The type of data: As declared before, RS concerns many application fields and makes use of a range of information from different backgrounds. This information can be presented as ratings, social relationships, contextual information, items' content...etc.

The recommendation model: Two main recommendation models were recognized in the state-of-the-art. One category is model-based and it consists of using the available data to learn the users' preferences and predict them. Another one is memory-based and it consists of using directly the available data for prediction.

The recommendation technique: RS techniques differ widely and concern the probabilistic models, bio-inspired models, and based statistical models.

The sparsity level: Sparsity is a parameter on which the recommendation technique is chosen and the scalability of the system can be measured.

The performance of the system: Some algorithms involve higher resource requirements than others. This factor should be taken into consideration during the design of a recommendation algorithm.

The objective of the system: Two main known objectives were considered in the state-of-the-art. The first objective concerns the rating prediction task while the second only consists of recognizing the relevant items.

The desired recommendation quality: Recommendation quality is a multi-criterion matter, whereas both recommendation relevancy and recommendation diversity are two conflicting quality metrics.

2.2.5 Information types

RS was deployed successfully in several domain applications and many techniques were developed to accomplish its role in assisting the users' choice making. As [Bobadilla 13] declared, the recommendation generation depends on the available data on which a suitable recommendation technique could be applied. The data used in a recommender system can be classified [Bobadilla 13] into a taxonomy. This taxonomy considers three data factors: (1) the target of the data: user or item; (2) the mode of acquisition: explicit (i.e., ratings to items made by users) or implicit (e.g., number of times a user has heard a song); and (3) the information level: memory, content or social context. This classification is further explained in the next items:

Behavioral information: This type of information depends on the acts of users in the system, whereas the users' actions are stored to model their preferences. As an example, for a user browsing a website, the system stores the clicks, the timing, and the browsing sequences. Furthermore, a user can express his preferences in a clearer manner by giving an evaluation. Behavioral information can be classified into two types [Jawaheer 14]:

- **Explicit information:** For example: ratings in binary range as on YouTube, or in a wider range as from 1 to 5 on Netflix.
- **Implicit information:** For example: comments, social actions like sharing a post on Facebook, time spent..etc.

Demographic information: This type of information is less used among the rest because of the privacy problem. Demographic information such as age, location, profession, and sex can be used in many cases to overpass RS limitations [Burke 07].

Social information: It concerns the relationships between users and it has many levels: family members, school or work colleagues, or just virtual online links such as Facebook friends [He 10].

Content information: Items under trade, in RS, have many forms and can be described by their content, whereas information extraction tools were applied on media resources or texts to represent each item in a more useful form [Lops 11].

Contextual information: Context has two main aspects (i.e., time and location) which have a major role especially in the tourism field. As an example, tourism applications depend on seasonal events. Moreover, road routing application needs time information to predict traffic quality..etc [Adomavicius 05b].

2.2.6 Recommendation algorithms

According to the information types presented in the previous section, different techniques were proposed. Mainly, there is three type of recommendation algorithms: col-

laborative filtering algorithms (CF) [Desrosiers 11], content-based filtering algorithms (CBF) [Lops 11], and hybrid filtering algorithms (HYB) [Burke 02b]. In the next sections, we discuss, in more detail, these techniques.

2.2.6.1 Collaborative filtering algorithms

Collaborative approaches predict the users' preferences by considering their ratings as a whole; using appropriate aggregation formulas. This approach takes inspiration from the fact that humans tend to like recommendations received from close friends (i.e., at the social level.) or similar individuals (i.e., in the concrete level.) than different individuals. This phenomenon was developed in the research field and yielded two categories of CF algorithms: memory-based collaborative filtering (MeCF) and model-based collaborative filtering (MoCF) [Ekstrand 11].

Model-based collaborative filtering: This family of techniques uses the available data to build a prediction model. To this end, many machine learning techniques were adapted and include classification techniques, bio-inspired techniques, and data reduction techniques.

Matrix factorization methods (MF): Matrix factorization was used in the collaborative algorithms to solve the scalability problem and to improve the relevancy of the recommendations. MF allows representing both users' profiles and items' profiles in the form of a numerical vector of latent variables. Ratings are, then, predicted by computing the product between the latent vector of the target user and the latent vector of the item. It exists four matrix factorization techniques: singular value decomposition, principal component analysis, probabilistic matrix factorization, and non-negative matrix factorization. MF mainly uses the available information about the users' ratings, social information, and more to target relevancy. Moreover, MF was used to solve the sparsity problem, whereas PCA was used to condense the rating matrix which consequently improves the efficiency of the similarity measures [Bokde 15].

Classification methods: For an object represented by a vector of characteristics, a classifier allows assigning this object to a labeled class. In fact, three classification types exist and are supervised classification, semi-supervised classification, and unsupervised classification. Among the classification techniques, we mention k-nearest neighbors classifier (k-NN), decision trees [Cho 02], association rules [Cho 02], support vector machine [Zhang 02], Bayesian networks [Park 07], and neural networks [Cheng 16]. These techniques were mainly used in the RS field to hybrid more than a recommender, to rank the recommendation set, and to predict the ratings.

Memory-based collaborative filtering: These techniques can be applied to the items' information (i.e., item-based collaborative filtering (ICF).) or the users' profiles

(i.e., user-based collaborative filtering (UCF).), whereas *k*-nearest neighbor algorithm (*k*-NN) is the main algorithm used to select for each target user/item *k*-nearest neighbors according to a similarity measure. Then, those neighbors can be used to predict the ratings using an aggregation formula. Finally, the recommendations can be identified by selecting the items having the highest rating prediction values.

Clustering: The clustering is an unsupervised classification technique that allows to group together similar instances based on their features. The basic idea in clustering techniques consists of maximizing the internal density of the composed groups by minimizing the pairwise distances between each pair of members and maximizing the distances between each pair of clusters' centers to separate them. In effect, the users/items are represented by sparse vectors of their ratings, whereas the similarity measures can be applied on the different vectors by considering the differences between the rating [Li 03] [Al Mamunur Rashid 06] [Xue 05]. These measures were used in two types of clustering algorithms:

- **(Statistics/probabilistic)-based clustering:** Algorithms such as *k*-means, fuzzy *c*-means, *k*-NN, and expectation-maximization (EM) algorithms were used for clustering. These algorithms benefit from the available information about the users to progressively create clusters of similar users. *k*-means is a clustering algorithm known for its suitability to large datasets, which makes it a good choice for RS applications. In particular, *k*-means selects randomly *k* initial centers or seeds, then progressively optimize the quality of the clusters by adjusting these centers [Dakhel 11].
- **Bio-inspired based clustering:** A main issue in the previous clustering techniques is their dependence on the quality of data, while this latter suffers the sparsity problem. Additionally, stochastic techniques such as *k*-means algorithm depend on the selection of the initial seeds, which is a critical step that affects the quality of the final recommendations. Including the bio-inspired techniques such as the genetic algorithms may allow overpassing this issue by focusing more on better quality indicators (e.g., precision, MAE..etc.), or by optimizing the quality of the initial seeds [Wang 14b][Georgiou 10].

2.2.6.2 Demographic-based filtering algorithms

Users' information such as age, sex, profession, and location were used for recommendation in demographic-based filtering algorithms (DEM). This information is valuable to alleviate the cold start problem, whereas the users' similarity is computed by counting the number of common demographic features. This similarity can be explained by real-life examples, whereas a user may like similar experiences that his geographical neighbor or a same-aged user likes [Wang 12][Safoury 13][Zhao 14] [Dai 09].

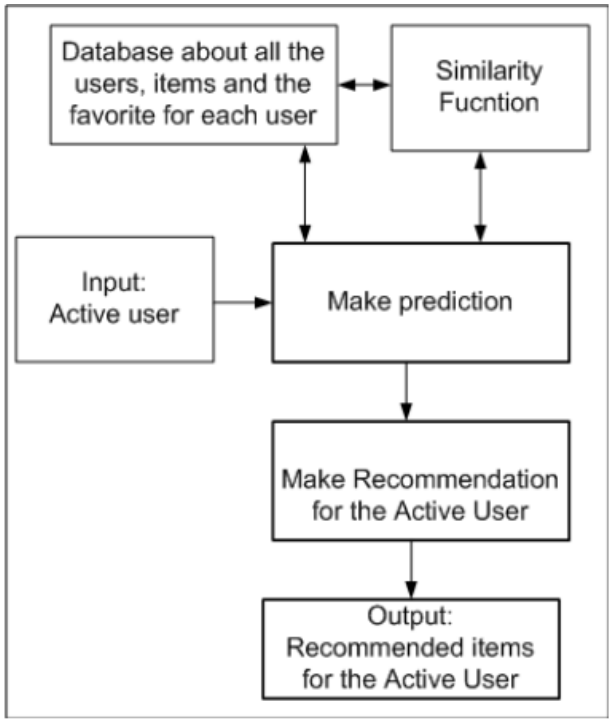


Figure 2.1: Flow of memory user-based collaborative filtering process.

2.2.6.3 Content-based filtering algorithms

Content-based filtering algorithms (CBF) are similar to demographic filtering, whereas the items are represented using a set of features extracted from their content (i.e., descriptions, meta-data, trailers, ..etc.). In the case of movies, each item can be represented by the set of genres, actors, producers, and year of release. Furthermore, textual descriptions, images, or thrillers can offer additional information after a suitable processing step. Content-based filtering tools allow measuring the importance of each feature and its correlation to other features. The acquisition of these features can be used to measure the similarity between different items, then generating the recommendations [Son 17].

2.2.6.4 Hybrid filtering algorithms

Hybrid filtering algorithms (HYB) appeared as a solution to overpass the individual limitations of the previously cited recommendation techniques. A common trend is to combine CF algorithm with content-based filtering or demographic filtering algorithm to overpass the cold start problem [Strub 16] [Paradarami 17] [Logesh 19a] [Kim 17]. According to [Burke 02a], it exists many hybridization models such as the next:

Weighting: Rating prediction is made by linearly combining scores given by several recommenders, whereas an importance weight is assigned to each recommender.

Switching: Switching allows selecting which recommender to use. This selection focuses on the state of the item or the user in terms of his/her profile length. As an example, cold users would better get recommendations from a content-based recommender.

Mixed: This method generates mixed recommendations set by merging the recommendations obtained from several recommendation algorithms.

Cascade: In the first step, a recommendation technique is used to generate a recommendation set. Then, a second recommender is used to refine the initial recommendation set.

Feature augmentation: A first recommender system is used to predict or to classify the users' preferences. Then, its output is used to expand the users/items features before applying another recommender.

Meta-level: In comparison to feature augmentation, this hybridization technique allows putting a trained prediction model as an input of another recommender system.

Feature combination: Features from the items and the users are combined together to form only one representation. This latter is used to predict the preferences of users.

2.2.6.5 Social-based filtering algorithms

Social information (SI) was mainly used in social collaborative approaches (SOC) as an additional input to increase the recommendation relevancy. SI exists in two forms: a) explicit as in the case of family members, whereas a user may purchase a product recommended by one of his family members. b) implicit as it can be inferred from the interaction frequency between two users, or their social actions. A classification of recommender systems based on social networks was proposed in [Yang 14] as in the following:

Matrix factorization based social recommendation approaches: The social effect allows improving MF by incorporating more data in the prediction model. This data improves the preferences described in the objective function, which increases the precision of the recommendations.

Neighborhood based social recommendation approaches: Social information was used in neighborhood-based filtering algorithms in two ways:

- **Social network traversal based approaches** In effect, a positive correlation between the users' similarity and their mutual trust was proven, whereas social information was exploited for clustering, neighbors selection, and rating prediction. The social links between the users can be explored in different ways such as

in random walk [Jamali 09], which allows to cover direct and indirect neighbors and amount their weight on the final rating prediction.

– **Nearest neighbor (NN) methods** The methods classified under this category create two types of neighborhoods. The first neighborhood can be created using social information and performing a breadth-first search algorithm [Demovic 13] to select similar neighbors. The second neighborhood can be created by selecting similar neighbors based on the classical similarity measures such as the Pearson correlation. Finally, the recommendations generated from both neighborhoods are combined in a final recommendation set.

2.2.6.6 Comparison of different recommendation algorithms

RS was applied in various applications, where several information types are available for use. In this context, it exists different recommendation algorithms which have dissimilar performances. In [Cacheda 11], the authors have raised two questions about the evaluation of an RS. In the next elements, we summarize these criteria:

"How should an evaluation be performed ?" An RS can be evaluated in two manners: online and offline. In the offline evaluation, the data are divided into two sets: a training set and a testing set. In this division, an equal distribution of the users' preferences should be respected. A common approach consists of taking around 70% to 90% of data for training and the rest for the test. Differently, the online evaluation does not require pre-defined data, because it directly offers recommendations to users, and queries their preferences.

"Which is the best algorithm (or algorithms) in a particular context ?" Most constitutions in the literature interest by comparisons to basic algorithms which they dominate. Context analysis concerns different application domains, different training conditions, different experimental methodologies, and a comparison of algorithms from the same categories.

In Table 2.1, we compare different recommendation algorithms in terms of recommendation quality, and their tolerance to sparsity, cold start, and scalability problems. We used a value of 1 to 5 stars to show the differences between the different algorithms, whereas each number of stars represents a superiority level. Additionally, we used the matrix factorization algorithm as a reference technique for its better performance in comparison to MeCF algorithms.

Algorithm	Diversity	Relevancy	Sparsity	Cold Start	Scalability	Simplicity	Example
ICF	⋆,1	⋆⋆⋆,2	⋆	⋆	⋆	⋆⋆⋆⋆	[Shambour 16]
UCF	⋆⋆,1	⋆⋆,2	⋆	⋆	⋆	⋆⋆⋆⋆	[Koohi 16]
SOC/DEM	⋆	⋆	⋆⋆⋆	⋆⋆⋆	⋆	⋆⋆⋆⋆,4	[Krulwich 97]
CBF	⋆,3	⋆⋆	⋆⋆⋆	⋆	⋆	⋆	[Salter 06]
HYB	⋆⋆	⋆⋆⋆⋆	⋆⋆	⋆⋆	⋆	⋆⋆⋆	[Porcel 12]
MF	⋆⋆⋆,1	⋆⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	⋆	[VOZALIS 07]

Table 2.1: Comparison of different recommendation techniques.

In the next items, we explain some specific differences between these algorithms:

(1): Among the different CF algorithms, UCF is one of the most methods that give better diversity due to the diversity of the users in the selected neighborhoods.

(2): ICF and UCF were compared to each other by changing the size of the neighborhood. In effect, ICF gave better precision when UCF uses a small size of the neighborhood, by growing up this parameter, UCF becomes far better. More details about both algorithms will be added in the next Chapter.

(3): Content was used for recommendation by measuring how many common content items exist between two items. However, the similarity selection-based approach hardly limits the diversity as the recommendation set would belong to one or a few categories.

(4): Social and demographic information was used for both MeCF and MoCF algorithims, especially in MeCF algorithms to improve the similarity measures between the pairs of users.

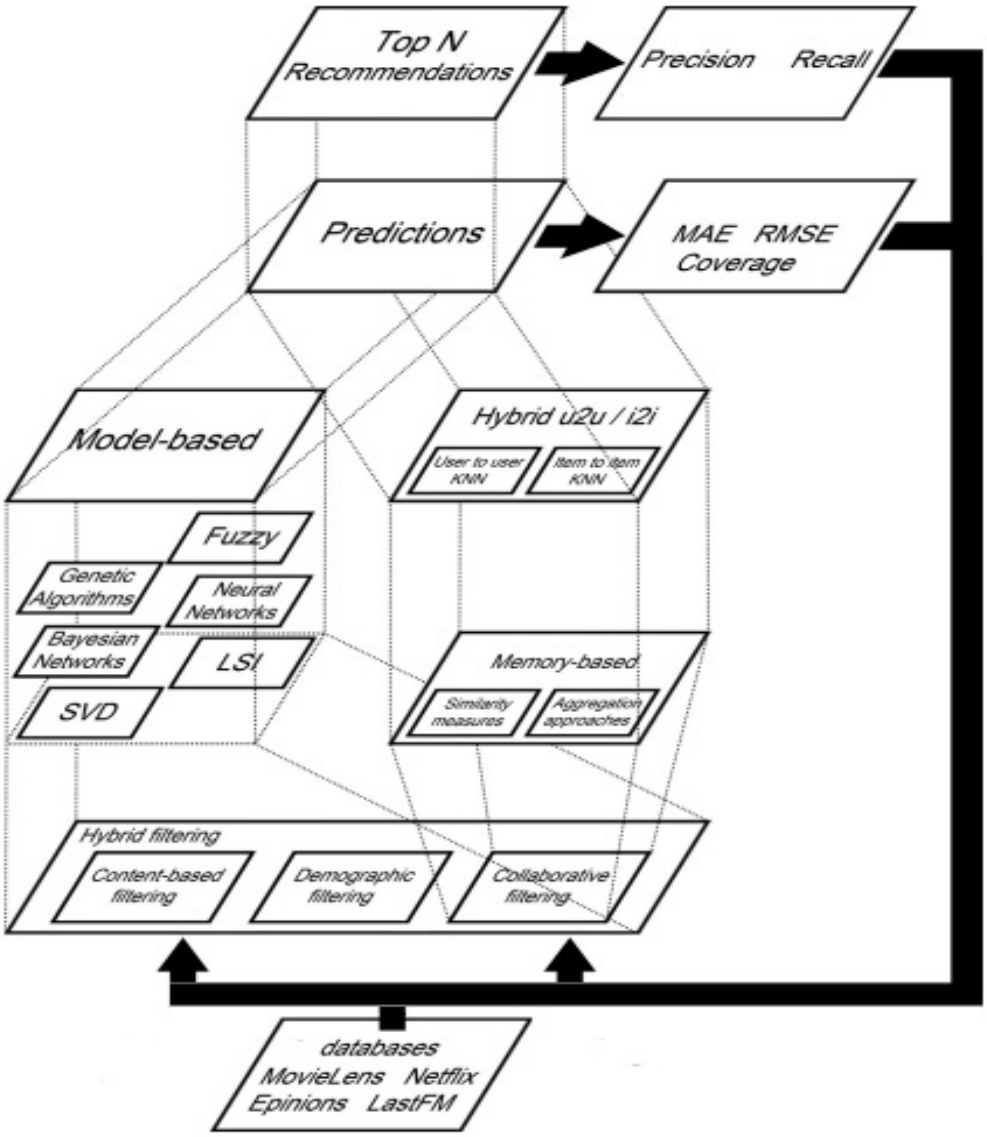


Figure 2.2: General representation of recommender systems development [Bobadilla 13].

2.2.7 Recommendation evaluation

Evaluating the recommendations allows measuring the efficiency of a recommender to face the needs of users. To this end, the research community has yielded to three evaluations categories [Shani 11] [Steck 13] [Gunawardana 09]:

2.2.7.1 Relevancy metrics

These metrics catch the accuracy of the recommender system at predicting a rating. These metrics belong to two different categories. In the first one, the metrics measure the averaged distance between the real ratings and their correspondent predicted ones. In the second category, the metrics measure the relevancy of the recommendations as a whole set. We list these metrics in the next items:

Rating prediction evaluators:

- **Mean average error (MAE):** MAE is the average error over all predicted ratings of all the users normalized by the number of the users and the number of the recommendations given to the users as in the next equation:

$$MAE = \sum_{u \in U} \sum_{i \in I_u} \frac{|R_{u,i} - R_{u,i}|}{|I_u|} \tag{2.2}$$

- **Root mean square error (RMSE):** RMSE is the same as (MAE), however the error is squared as in the next equation:

$$RMSE = \sum_{u \in U} \sum_{i \in I_u} \frac{(R_{u,i} - R_{u,i})^2}{|I_u|} \tag{2.3}$$

Recommendations set evaluators:

- **Precision:** It counts the portion of relevant items among the whole recommendation set and over all the users, as in the next equation:

$$Precision = \sum_{u \in U} \sum_{i \in R_u} \frac{|\{i|R(u,i) >= \theta\}|}{|R_u|} \tag{2.4}$$

- **Recall:** It counts the portion of relevant items in the recommendation set against all relevant items the user has expressed. This metric is introduced in the next equation.

$$Recall = \sum_{u \in U} \sum_{i \in TEST\_SET_u} \frac{|\{i|R(u,i) >= \theta\}|}{|TEST\_SET_u|} \tag{2.5}$$

- **F1:** It is the harmonic average of Recall and Precision, whereas a value equal to 1 is the best, and 0 otherwise. It is defined as in the next formula:

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{2.6}$$

2.2.7.2 Diversity metrics

Diversity: Diversity of recommendations means the averaged distances between all items in the recommendation set. This distance can be measured on the basis of the held content or the ratings assigned to an item. The next equation defines the recommendation diversity.

$$Diversity = \sum_{u \in U} \sum_{(i,j) \in R_u} \frac{(1 - sim(i,j))}{C^2_{|R_u|}} \tag{2.7}$$

where, $sim(i,j)$ is the distance between the items i and j, and $C^2_{|R_u|}$ is the number of all possible item pairs.

Novelty: This metric measures the novelty of an item in comparison to the rest of the items or to a target user. The novelty of an item can be computed by averaging the distances between a given item and the set of items in a user's profile, or by normalizing the number of ratings given to the item by the number of users. The next equations introduce both of these definitions:

$$Novelty = \sum_{u \in U} \sum_{i \in R_u} -log(\frac{|U_i|}{|U|}) \tag{2.8}$$

$$Novelty = \frac{1}{|I_u| \times |R_u|} \sum_{u \in U} \sum_{i \in R_u} \sum_{j \in I_u} sim(i,j) \tag{2.9}$$

Coverage: Recommendation coverage can be calculated by counting the number of recommended items against the whole number of possible items. This metric is considered a system evaluation metric because of its effect on commercial revenues. It can be defined as in the next equation:

$$Coverage = \sum_{u \in U} \sum_{i \in R_i} \frac{|\cup R_u|}{|I|} \tag{2.10}$$

2.2.7.3 Comparison of evaluation metrics

Recommendation quality can be validated in terms of several variables such as the distance between the real ratings and the predicted ones, the portion of relevant recommendation in respect to the recommendation list size, and the order of the recommendations (i.e., according to the novelty, diversity, relevance...etc). Another factor that can be considered is the benefices of the recommendation set on the target user or the system. As an example, a big recommendation coverage value reflects a better promotion of the system's products. We summarize in Table 2.2 these differences.

Metric	system	user	recommendation set	recommendation rank	prediction error
MAE	-	×	-	-	×
RMSE	-	×	-	-	×
Coverage	×	-	×	×	-
Diversity	×	×	×	×	-
Novelty	×	×	×	×	-
Recall	-	×	×	-	-
Precision	-	×	×	-	-
F1	-	×	×	-	-

Table 2.2: Comparison of different recommendation evaluation metrics.

2.2.8 Recommender systems tasks

As we design a recommender system, we address the issues that concern the needs of the users and the system. In [Herlocker 04], the authors analyzed the state-of-the-art and reviewed RS designs to identify the next eleven tasks of an RS:

Find some good items: A typical task of an RS is to identify within a long set of items, relevant ones to a user. Some RS offer also rating predictions of the recommended items, as an indicator of how much an item may be liked by the target user. Among these items, some may be irrelevant to the users' preferences because of the recommendation fatality.

Find all good items: In some contexts, it is so important to identify all relevant items within a low number of possibilities. On the contrary to typical RS applications, a lawyer looking for all correspondent cases of his current case can be a good illustrative example.

Annotation in context: In this case, the RS task is to comment, explain, or justify a recommendation by annotating and showing why it worths being recommended.

Recommend a sequence: RS recommends a sequence of items instead of only one item. Typical examples can be successive music tracts, a book that requires some previously read books, or a touristic trip that should be taken in successive steps.

Recommend a bundle: RS recommends a set of items that fit together as a group. A simple example can be a set of food products which should be bought together.

Find credible recommender: RSs are not always considered reliable systems by the users. Thus, some users tend to examine the efficiency of the system by entering the preferences and asking for recommendations to see how much they like it. Some RS allows this option by facilitating the evaluation of the recommendations.

Improve the profile: Some users tend to express more preferences to improve their profiles. These users believe that giving more information improves the quality of the recommendations which they will receive from the system.

Express self: MovieLens' power users were interviewed by [Herlocker 04], in which they conducted that some users keep expressing preferences just because it felt good.

Help others: Differently from the previous item, some users express preferences with an intention to help others because they believe that adding more preferences will help the system to generate more reliable recommendations.

Influence others: This is a negative act in RS applications, whereas a set of users may express high ratings to particular items to push others to purchase them.

2.2.9 Recommender systems applications

In the real world, RS was applied successfully for different purposes and on diverse levels. Big companies such as Facebook, Google, Amazon, and Netflix have developed their RS. These recommendations specifications vary from book trading, movies, social networks, tourism, to web content personalizing. In the academic field, we notice the existence of many benchmarks available for academic research. GroupLens [Harper 16] has offered various data sets for research. These resources differ in terms of their size, the covered domain, and the diversity of information that it holds. In this section, we enumerate both Movielens 100k and Movielens 1M as reference data sets which we used during the preparation of this thesis.

Movielens 100k: This data set composes 943 users associated with their demographic information and 1682 movies with descriptions, whereas each user has at least 20 rating. It consists of 100.000 ratings collected during 9 months of activity. The sparsity level of this data set is 6%. Movielens 100k is suitable for initiative academic research due to its low resource requirement.

Movielens 1M: This data set is composed of 6040 users associated with demographic information and 3952 movie with descriptions, whereas each user has at least a 20 rating. It consists of 1.000.000 ratings collected during 2 years and 0 months of activity. The sparsity level of this data set is 4%. Movielens 1M is suitable for simulating real-world applications and studying the scalability effect because of its suitability for experimental work.

In addition to Movielens 100k and Movielens 1M, GroupLens has released many other movie data sets of different sizes such as MovieLens 10M and MovieLens 20M. On October 2, 2006, Netflix [Bennett 07] has released a competition to improve its recommender' s precision by 10%. Netflix has offered another time a movie benchmark that consists of 480,189 users, 17,770 movies, and 100 million ratings. This data set is considered the biggest available benchmark for research.

We also enumerate the next data sets which cover RS applications from different aspects:

Jester: A Joke data set collected between April 1999 - May 2003 from the jester.com website. It contains 4.1 million real value ratings of 100 jokes from 73,421 users. Every user has rated at least 36 jokes on a scale between -10 to +10.

Lastfm: Last.FM API has been used to collect a set of public information about music. It contains two types of data: tags and similar songs. It also holds the next features: 505,216 tracks with at least one tag, 584,897 tracks with at least one similar track, 522,366 unique tags, 8,598,630 (track-tag) pairs, and 56.506.688 (track - similar track) pairs. It exists a public recommendation dataset [Celma 10] from this source, formed of 1000 users, and 19 million ratings collected till May, 5,th,2009.

Book-Crossing: Books recommendation data set composed of 278,858 users associated with their demographic information who have given 1,149,780 ratings to 271,379 books. Ratings are given on a scale from 1 to 10, whereas 1 means a bad evaluation and 10 is a good evaluation.

The availability of different recommender system data sets for academic research, confirms the importance of such a field and its impact on real life.

2.3 Recommender systems: issues and limits

RS development is faced with a number of issues that limit its efficiency [Sharma 13]. From one side, the few and sparsely available information about the items and users decrease the ability to train reliable and efficient prediction models. On the other side, the users' preferences are not fully observed, whereas only a few users tend to explore novel and diverse items while much more others only seek to get relevant recommendations. In the next elements, we detail these issues separately with more explanation.

2.3.1 Cold start problem

The cold start problem is related to the situation when new users or new items are added to the system. In the case of memory user-based collaborative filtering, the neighbors of a cold user can not be identified efficiently because of the few numbers of co-rated items. In general, RS tends to directly ask the user to express more explicit preferences, or it can, instead, focus on different information such as contextual or demographic information. Similarly, cold items can not be recommended until having enough evaluations. This problem has two impacts on a RS: a) the recommendation's relevance decreases. b) the recommendation's coverage decreases [Lam 08] [Lika 14].

2.3.2 Data sparsity

Data sparsity is referred to as low preference expression face to the high number of items and users. The sparsity level is computed by the total number of possible preferences in RS minus the available number of preferences. The main difference to the cold start problem is that data sparsity implies the existence of the information about the users or the items but in a way that makes the neighborhood selection a hard and inefficient task because of the unequal distribution of evaluations [Nilashi 18] [Sarwar 00].

2.3.3 Scalability problem

RS applications have grown up to scale up a high number of users such as Facebook, which has 2.32 billion active users monthly. This number of users involves solutions to speed up information processing and recommendation generation. To this end, two types of techniques were applied:

Material solutions: In the basic level, graphical processing units (GPU) have been adapted to process a bigger amount of information in recommender systems. Furthermore, parallel architectures and distributed systems were proposed as solutions to dividing the recommendation task into subtasks. In this context, the Map-Reduce paradigm took place as an efficient approach to manipulate complex problems [Yu 12] [Zhao 10] [Sarwar 02a].

AI solutions: These solutions include based-static and based-bio-inspired clustering techniques, in addition to classification and data reduction techniques. The basic idea consists of reducing the data dimensionality to alleviate its processing complexity. Clustering techniques were used to group in smaller groups users or items together, with the aim of reducing the number of required computations needed every time the recommendations are generated. Based-statics data reduction techniques such as PCA, LSA, LDA, and SVD were used as well as supervised and unsupervised data reduction solutions [Sarwar 02c] [Nilashi 16a]

Improving the scalability of the systems allows enhancing the quality of the recommendations as well by dividing the data into simpler and more suitable subgroups. Additionally, it speeds up the recommendation generation.

2.3.4 Recommendation redundancy/popularity

The recommendation quality was evaluated only on the basis of the relevancy criteria. However, a novel recommendation suggests that the novelty and the diversity of the recommendations are equally important quality criteria. Studies in the literature show a negative correlation between the recommendations' relevancy and their novelty/diversity. The different contributions aimed to balance these conflicting objectives

by targeting a trade-off due to the positive effect of recommendations diversity/novelty on satisfying the users and improving the commercial revenues as well [Park 08].

2.4 Recommender systems: trends

RS evolution tended to progressively integrate an increasing amount of data to achieve a higher prediction accuracy. This evolution was accompanied by the evolution of the web content which passed from web 1.0 to web 3.0. According to [Bobadilla 13], RSs passed by three evolutions:

(1) At an earlier age, RS has only used explicit preferences, demographic information, and content information.

(2) By the appearance of web 2.0, social information was considered in addition to the previously mentioned information. Friends, followers, and both trusted and untrusted relations were as well used to enrich the RS. Additionally, blogs, tags, comments, photos, and videos were exploited to better model the users' behaviors.

(3) By the appearance of web 3.0 and the internet of things, the information obtained from devices and sensors was integrated and exploited into the so-called context-aware recommenders. The new integrated information knew a success, particularly in the tourism field.

In the next sections, we detail two well-known categories of recommender systems that appeared with the evolution of the web.

2.4.1 Contextual recommender systems

Context-aware recommender systems integrate signals received from devices such as time, and GPS location to improve the users' behavior modeling [Adomavicius 10]. In particular, geographic RS has appeared as an exploitation of the new availability of mobiles information which can be associated with the next chronology of RS development:

Conventional RS (RS): Only classical information was used, like behavioral information, content information, demographic information ...etc.

Geographical recommender systems (GRS): In this type, recommendations can be generated using geographical information. A typical example is recommending restaurants based on location, time, and likeliness of similar clients. Similar applications include cinemas, supermarkets, cultural activities... etc.

GRS+: In this type, a rating is given to an item according to the distance between a target user and an observed item. Thus, three possibilities can be established:

1. **Hybrid CF/DEM filtering:** each item can get only one vote from each user with respect to the distance between the user and the item.

2. **Geographic RS:** each item can get more than one vote according to the position/location of the user.

3. **Extended CF/DEM filtering:** in which the geographical information is given to each vote instead of giving it to each user.

2.4.2 Bio-inspired algorithms in recommender systems

2.4.2.1 Different uses of bio-inspired algorithms

Evolutionary algorithms were used differently over recommender system contributions. A survey [Peška 19] about using swarm intelligence (SI) cites various applications of SI in RS algorithms:

Feature weighting in RS: selecting features in RS fields concerns context-aware recommenders, content-based recommenders, and hybrid recommenders. These weights may take binary values (i.e., 0 or 1.) or real values (i.e., between -1 and 1.).

- **Context aware recommenders:** SI used contextual information to improve the recommendation quality by combining or selecting contextual features using a binary formulation of the selection problem [Gupta 17]
- **Nearest based models:** SI was used to improve the similarity between users, by considering additional data sources or by incorporating trust-based similarity basics [Rad 07].

Clustering: This technique was applied in RS to improve the scalability and alleviate the cold start problem. SI-based clustering was combined with statistical-based clustering techniques in several works [Kim 08] [Bobadilla 11b]. However, the authors of [Peška 19] note the miss of enough evaluation and description of SI-based clustering applicability to RS in comparison to other bio-inspired algorithms.

Graph based recommenders: user-user similarities or item-item similarities can be represented in a graph by creating weighted edges between each pair. This graph was exploited in RS field using SI techniques for recommendation purposes, whereas two problems were concerned: recommending a user to another user, and mining trust between users [Bedi 15] [Sherkat 15].

- **Trust based recommenders:** SI was used to select trustful neighbors, on which a reliable recommendation can be performed [Kaleroun 14] [Bedi 12].
- **Social network-based recommenders:** In the social context, SI-based recommenders were proposed for recommending friends [Bedi 15] [Sherkat 15].

Re-ranking and ensemble approach : SI was used to combine different recommendation techniques by assigning to each one an importance weight. Also, it was used to target a multi-objective re-ranking problem [Chifu 15] [Katarya 17].

Latent factor models: SI and GA algorithms were combined or incorporated with matrix factorization algorithms in many ways. [Kagita 14] [Salman 16].

2.4.2.2 Frequent bio-inspired algorithms used in recommender systems

By reviewing the state-of-the-art, in particular, the surveys in [Peška 19] and [Alsarhan 18], we could enumerate the use of the next bio-inspired algorithms:

Using genetic algorithms (GA): Genetic algorithms adopt evolution principles of natural selection and survive for the best. GAs were used for optimization and search problems using genetic operators such as mutation, crossover, and selection. The quality of obtained solutions in each step is measured by a fitness function [Bobadilla 11b] [Kim 08].

Using ant colony optimization (ACA): This probabilistic technique can be applied to problems such as identifying the best path within a graph. It is inspired by the way an ant moves to food sources and comes back to its nest. In each movement, the ant reinforces the amount of pheromone in the path it passes by, which influences the behavior of other ants looking for food. As the pheromone amount decreases over time, an artificial ant considers, in its movement from a node to another one, two probabilities: the cost of the path and the amount of the current pheromone [Sobecki 10] [Gohari 17].

Using particle swarm optimization algorithms (PSO): The algorithm is inspired by birds' flock behavior. Each particle is defined by its local best solution, the global best solution, and its own velocity. In each iteration, particle velocity is updated toward the best solution by considering the current velocity and the global and local best solutions. After velocity update, the best solutions are updated as well [Alam 12] [Katarya 16].

Using neural networks algorithms (ANN): These algorithms are inspired by the animal brains, whereas a set of artificially connected nodes in many layers. (i.e., in various architectures.) mimics the way in which the brain performs. Each node has a set of entries with a weight for each entry and an activation function. ANN learns by propagating information from the nodes of the first layer to the nodes in the last layer and comparing the output of the network with the correct output, then adjusting the weights of the network to fit the existing patterns [Paradarami 17] [Kumar 17].

Using artificial bee colony (ABC): This algorithm mimics bee colony, whereas bees are classified into employed bees, onlooker bees, and scout bees. Employed bees are initially assigned to solutions that are improved progressively by crossover with other solutions. Onlooker bees select randomly one solution and seek to improve its quality by looking in the surrounding. If the solutions are not improved after a number of tries, the bees become scout bees and new solutions will be assigned randomly [Katarya 18] [Yadav 18b].

Using bat algorithm (BAT): The algorithms are inspired by the echolocation of bats with varying loudness and emission of pulse rate. As a bat gets closer to prey, the

loudness decreases, and the pulse rate increases. BAT consists of an explorations step and an exploration step similarly to PSO [Yadav 18b] [Yadav 18a].

Using fish school search (FSS): Similarly to PSO algorithms with more extensions, whereas each fish has a current state that does not change till finding a better solution. In each step, each fish perform two movements: collective-instinctive movement and collective-volitive movement [Rozie 14] [Zhang 18].

In table 2.3, we differentiate the different using of bio-inspired algorithms in RS.

Algorithm	GA	ACO	ANN	ABC	BAT	FSS	PSO
Similarity	X	X	X	-	-	-	-
Weighting	X	-	X	-	X	X	-
Clustering	X	-	-	X	X	X	-
Re-ranking	X	-	-	-	-	-	X
Latent factor models	-	-	-	-	-	-	X
Graph-based models	-	X	-	X	-	-	X

Table 2.3: Summary of different use of bio-inspired algorithms in RS.

2.4.2.3 Evaluation of evolutionary algorithms

Many bio-inspired techniques were proposed in the literature. These techniques differ in terms of their convergence speed, and the quality of their solutions. In [Peška 19], the authors listed a set of factors on which an evolutionary algorithm can be evaluated:

Strengths: EAs are good for multi-objective qualities, and black-box functions, as well as optimizing complex recommendations scenarios.

Weaknesses: EAs suffer time complexity, parameter issues, scalability, and reproducibility problems. Additionally, the existing EA-based contributions only concern outdated recommenders.

Opportunities: EAs can be successfully integrated into complex recommendation models, used for combining different recommendation qualities, or targeting the users' and the system's benefits at once.

Threats: It is notable that graph-based and context-based recommenders achieve better performance in some scenarios. Furthermore, factorization techniques and deep learning have recently got more focus.

2.4.2.4 Evolutionary algorithms through the state-of-the-art

According to [Peška 19], SI algorithms can be characterized by a set of features that we can generalize to other EA. In the next, we enumerate these characteristics:

Baselines: The survey in [Peška 19] notes that half of SI-based recommender systems papers published till 2019; were compared only against memory-based collaborative filtering (MeCF) algorithms.

Evaluation metrics and methods: The most adopted quality criterion is accuracy, while novelty, diversity, and coverage were considered in less frequency.

Scalability and time complexity: SI-based algorithms are time taking, whereas most papers have used in their experimental work small-scale datasets. In fact, a dataset that contains 1 million ratings is considered a large-scale dataset.

Reproducibility Most papers were performed on private datasets which limits the ability to reproduce them. In some other cases, the provided description is not enough to reconstruct the recommendation model.

Novelty SI-based recommenders are often based on incremental research, whereas the novelty of some papers concerned the application of EA to some specific issues in RS.

Threats Current tendencies are about deep learning and matrix factorization which knew considerable achievements in RS field, whereas EA was used only for pre-processing or post-processing the prediction model.

Positive aspects and opportunities EA is good for complex problems with many constraints. EA allows to generated diversified solutions due to its stochastic approach.

2.5 Conclusion

Recommender systems have benefited from the theory and applications deployed while developing the information retrieval field. However, these systems differ from search engines (i.e., which are a major application of the information retrieval field.) in their ability at modeling the users' preferences and offering relevant recommendations. The efficiency of recommender systems depends on many axes which are resources and quality requirements. From one side, the rapid growth of the web accompanied by the increasing users implies more work on improving the scalability. On the other side, the recommendation quality definition is changing. So, a contradictory multi-objective is targeted. Novelty, diversity, and accuracy are the keys to the new recommendation fashion. Currently, the users seek to get more than relevancy. A pleasant experience would hold some of the novelty and diversity in the recommendations set. In fact, recommender systems were used for both desktop and mobile applications, whereas many techniques have appeared for diversification purposes such as clustering and recommendation ranking. This latter has become an interesting technique due to the issue of hard content exploration. In Chapter 3, we present in more detail the MeCF and the recommendation process. Also, we present and discuss the recommendation

diversification techniques and the possible issues which can be faced while deploying these recommendation algorithms.

It is possible to invent a single machine which can be used to compute any computable sequence.

Alan Turing

3

Memory-based collaborative filtering approaches

▷ *In the previous chapter, we presented recommender system algorithms. This chapter gives more details about collaborative-based filtering algorithms and covers the recent contributions that target various limitations, specifically the scalability and the low recommendation quality. But first, we define this type of algorithms, and how it can be performed to generate recommendations and to rank them. Additionally, we present possible post-processing techniques that may improve the quality of the recommendation.* ◁

Chapter outline

3.1 Introduction

Collaborative filtering algorithms (CFs) have considerable importance in the RS field due to their results, simplicity, and ease of deployment [Schafer 07]. These algorithms are classified into two groups: MeCF algorithms and MoCF algorithms. In MeCF algorithm, recommendation generation passes by several steps: similarity calculation, neighborhood selection, rating prediction, and recommendation ranking. In the next sections, we delve deeper into these details. Additionally, we present the possible issues and analyze the different previous techniques to overpass them. Particularly, we give a more focus on diversification techniques, in the last section, which is the main interest of this thesis.

3.2 Collaborative filtering approaches

CF algorithms [Schafer 07] get their name from the prediction approach that consists of considering users' preferences together as a whole or a sub-group of users to infer how probable a target user may like an item. In the literature, collaborative algorithms are categorized into two types: MoCF and MeCF filtering [Bobadilla 13]. These types are further explained in the next sections.

3.2.1 Model-based collaborative filtering

MoCF consists of training a prediction model by feeding a machine learning technique with users' preferences data. MoCF algorithms create for each user a prediction model based on its previous preferences. This category of algorithms includes deep learning algorithms and matrix factorization algorithms.

3.2.1.1 Deep learning

Deep learning (DL) has known success in supervised and unsupervised learning tasks [Zhang 19]. DL is known for its ability to learn deep representation or multi-data abstractions. Technically, DL uses a variant of stochastic gradient descent to optimize a differentiable objective function. DL covers several architectural paradigms such as Multi-layer Perceptron (MLP), Conventional Neural Network (CNN), Recurrent Neural Network (RNN), and Restricted Boltzmann Machine (RBM). DL's importance in the RS field lies in its ability to maximize the efficiency of data representation and enhance the efficacy of multi-model techniques. These benefits can be summarized in the next items [Zhang 19]:

Nonlinear Transformation: DL can model non-linear patterns in user-item interactions, differently from matrix factorization, and machine factorization techniques.

Representation Learning: In real-world applications, a variety of available data can be exploited to improve the efficiency of the trained models. DL allows us to represent, weigh, and extract useful representation from heterogeneous data. DL can help in two ways: a) Facilitating data pre-processing tasks. b) Extracting significant representations from various types of data.

Sequence Modelling: DL was used successfully for natural language processing, voice recognition, and machine translation. DL can be exploited for time-aware recommender systems, whereas applications such as next-item prediction and session-based recommendation can benefit from sequence modeling to enhance their efficiency. DL is useful for modeling users' temporal behavior and items' evolution.

Flexibility: Technically, DL is a well-established domain, whereas many frameworks were deployed for its applications.

Although the numerous DL' benefits, some drawbacks can be summarized in the next items [Zhang 19]:

Interpretability: DL's output is hard to explain, whereas justifying the reason for recommending a given item is difficult. In ICF, a recommended item can be justified by the similarity between this item and another item already liked by the user, however in DL making justifiable recommendations is a more complex task.

Data Requirement: DL requires an intensive amount of data to accomplish its effectiveness. In real-world applications, users, and items number are in terms of millions. However, modeling this amount of data needs more storage and computation resources.

Extensive Hyperparameter Tuning: DL requires tuning many parameters, which is a common issue in the machine learning field such as in fixing the learning rates...etc.

3.2.1.2 Matrix factorization

Matrix factorization models [Bokde 15] allow uncovering the latent features of users' preferences. The main difference between explicit features and latent features can be explained using a real example. In the film industry, a user may prefer a specific movie due to its actors, or specific genres such as in the example (genre1 = action, genre2=romance, ...etc, actor1=Brad Pitt, actor2=Leonardo DiCaprio, ..etc). These explicit characteristics are not steady indicators about the user's preferences, who may prefer a given movie for more specific factors (i.e., which are not related to its genre, actors...etc) but a correlation between these different factors together.

Examples of latent features modeling include Probabilistic latent semantic analysis, neural networks, latent Dirichlet allocation, and matrix factorization. This latter has known a recent development due to its considerable accuracy and stability. Singular value decomposition (SVD) [Golub 71] was well established in the information retrieval field, however, its use for RS is limited because of the sparsity problem. The

first adaptation of SVD consists of using imputation methods, which fill the missing ratings using statistical methods. However, this technique increases the data complexity. A second approach consists of only considering the observed ratings.

Basic matrix factorization: Principal component analysis (PCA) [Nilashi 16b] is the simplest matrix factorization method, which allows extracting significant features from large dimensionality representation. PCA generates a reduced matrix of the original data by keeping only the top first dimensions with the largest variance. SVD [Patra 19] is a related matrix factorization, which allows reducing the original data into one common base of significant dimensions, whereas items' latent features q and users' latent features p can be represented by a matrix of lower rank as the rating prediction can be calculated as:

$$r(u,i) = \mu + b_i + b_u + q_i \cdot p_u, \tag{3.1}$$

where, μ is the average rating of users, b_i and b_u are the observed deviation of user u and item i, q_i, p_u are latent features of user u and item i. SVD tries to optimize the next objective function:

$$min_{b,q,p} \sum_{(u,i)\in K} (r(u,i) - \mu - b_i - b_u - q^T \dot{p}_u) + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{3.2}$$

This objective can be reached using a stochastic gradient descent algorithm, whereas, in each iteration, a parameter is optimized while the rest of the parameters are fixed as in the next rules:

$b_u \leftarrow b_u + \eta$(e(u,i)- $\lambda \cdot b_u$)

$b_i \leftarrow b_i + \eta(e(u,i) - \lambda \cdot b_i)$

$q_i \leftarrow q_i + \eta(e(u,i) \cdot p_u - \lambda \cdot q_i)$

$p_u \leftarrow p_u + \eta(e(u,i) \cdot q_i - \lambda \cdot p_u)$

where, η and λ are the learning rate and the regularization weights, which can be personalized for each variable separately by embedding more factors.

Matrix factorization with contextual information: SVD model has been extended in the SVD++ model [Koren 10], which incorporates the "implicit" information such as

the prior interaction between the user and the items (i.e., observed, seen,...etc.). SVD++' objective function can be described as the next:

$$r(u,i) = \mu + b_i + b_u + q_i^T(p_u + x|R(u)|^{\frac{-1}{2}} \sum_{j \in R(u)} y_j) \tag{3.3}$$

where, $R(u)$ is the set of items seen by user u, and y_j is the rating given by u to j. Time was incorprated in SVD++ model for the next reasons: a) users' biases change over time. b) items' biases change over time. c) users' preferences change over time. d) items' characteristics do not change over time. In the light of these facts, the users and the items biases were adapted as in the next formulas:

$$b_i(t) = b_i + b_{i,Bin(t)}$$

$$b_u(t) = b_u + \frac{\sum_{l=1}^{K_u} e^{-\lambda|t-t_l^u|} b_{tl}^u}{\sum_{l=1}^{K_u} e^{-\lambda|t-t_l^u|}} + b_t^u$$

In case of items' biases modeling, t is a day associated with an integer $Bin(t)$ in the range [1-30]. However, users' biases modeling is more complex, and involves k_u time points defined by:$\{t_1^u, t_1^u ... t_{ku}^u\}$ spaced uniformly, while b_{tl}^u parameters are associated with these time points and are automatically learned from the data.

Matrix factorization with social information: users' social information was embedded in several matrix factorization models, whereas the appearance of the probabilistic matrix factorization model [Mnih 08] has given birth to many social-aware MF recommenders such as SoRec [Ma 08], RSTR [Chen 13], and SocialMF [Jamali 10]. Mainly social-aware techniques exploit the social strength to model the similarity between the users, or to study its effect on the behavioral preferences. In this section, we take TrustMF [Yang 17] algorithm as an example to give more details about this type of approach. In TrustMF, a trust matrix T is of two dimensions having the same length as the number of users. In this model, users' trust is not reciprocal (i.e., we say that T is not symmetric). Also, the users who trust other users are modeled using a matrix of latent features B. Similarly, the users trusted by other users are modeled using another matrix of latent features W. Since trust features are partially observed, a matrix factorization with the next objective function is used to approximate a deduction model.

$$min \sum_{(u1,u2) \in t} (b_{u1}^T w_{u2} - T_{(u1,u2)})^2 + \lambda(||W||_F^2 + ||F||_F^2) \tag{3.4}$$

As the users set involved in the trust matrix T, and the rating matrix R is the same, one matrix factorization model can be used for both: approximating the users' preferences, and the users' trust relations. For the items' latent features matrix V, the final objective function of TrustMF is defined as:

$$min \sum_{(i,j)\in R} (b_i^T v_j - R_{(i,j)}) + \lambda_T \sum_{(u1,u2)\in t} (b_{u1}^T w_{u2} - T_{(u1,u2)}) + \lambda(||B||_F^2 + ||V||_F^2 + ||W||_F^2) \tag{3.5}$$

3.2.2 Memory-based collaborative filtering

MeCF algorithms use the rating matrix to predict the users' preferences. This latter is computed using an aggregation formula that uses the rating information. MeCF filtering was applied to users' profiles (i.e., user-based collaborative filtering.) [Bellogín 13], items' profiles (i.e., item-based collaborative filtering.) [Sarwar 01], and also by the combination of both [Wang 06]. In the next sections, we give more details.

3.2.2.1 Memory user-based collaborative filtering

User-based collaborative filtering algorithms (UCF) generate recommendations by selecting for every target user, a set of like-minded neighbors. Then, an aggregation formula is used to infer from the profiles of these neighbors, relevant items that the target user has not seen yet. The neighborhood selection is based on the similarity measures (i.e., can be considered as the distance between two users), and it depends on the co-rated items. If we consider the case of user u_1, defined by his profile $I_{u1} = \{i_1, i_2, i_3, i_4\}$, and a set of candidate neighbors u_2 and u_3 associated with their profiles $I_{u2} = \{i_1, i_2, i_3, i_4\}$ and $I_{u3} = \{i_1, i_2, i_3, i_4\}$ respectively. The similarity calculation between u_1 and u_2 considers only the items i_1 and i_2. Similarly, the similarity between u_1 and u_3 only considers the items i_1, i_2 and i_3. It exists many similarity measures that differ in their characteristics, their suitability to the employed data, and their effect on the quality of the recommendations. If we assume that u_2 is selected as a neighbor, then $I_{u2} - I_{u1}$ is the set of items subject to the recommendation. In general, the parameters which should be explored in UCF are the number of neighbors, the selection criteria, and the choice of the prediction formula.

3.2.2.2 Memory item-based collaborative filtering

In contrary to the previous method, item-based collaborative filtering (ICF) generates the recommendations by analyzing the similarity between items. In other words, if we have an item i liked by a user u, that means similar items to i would probably be liked as well. The application of this idea in RS applications on an unseen item j defined by a user u_1, consists of selecting a set of similar items from I_{u1}. The similarity between items is calculated similarly to the users' profiles, whereas only co-ratings are considered. If we take an example of item i_1 defined by its profile $\{u_1, u_2, u_3, u_4\}$ and another item i_2 defined by its profile $\{u_1, u_2, u_3, u_4\}$, only the ratings given by the common users are considered during the similarity calculation.

3.2.2.3 User-based versus item-based recommendations

A flexible comparison of UCF and ICF consists of taking into consideration several requirements such as the quality of the recommendations and the efficacy of the system. In the next elements, we cite from [Desrosiers 11], a list of the most influenceable factors in memory-based collaborative filtering approaches.

Accuracy: One can interrogate, what is a better algorithm to perform between UCF and ICF. As the collaborative approaches consist of using a similarity measure to select a neighborhood, the portion of the number of users and the number of items plays an interesting factor in predicting accurate recommendations. In the case where users are fewer than items, it is more preferable to perform UCF, because it is more probable to have wider co-rated items between the users, hence a better similarity. Otherwise, ICF is preferred.

Efficiency: In collaborative filtering, the similarities are computed between each pair of users/items, which emphasizes a correlation between the number of users/items and the efficacy of the system. In case when the users' number is fewer than the items' number, it is more preferable to perform UCF. Otherwise, ICF is preferred.

Stability: In real-world applications, a continuous stream of data is added every time, whereas new items, users, and ratings are inserted every while. A stable system should be able to make recommendations with the least required adaptation. In the case where new users are constantly added, as in the example of deploying a new recommendation application, it is more preferred to use ICF. In stable systems, items are instead constantly added, hence using UCF is preferred.

Justifiability: ICF is more suitable for justifiability since it allows the user to know, the neighbors participating in the recommendation process, which is not the same case in UCF.

Serendipity: In ICF, the recommendations are generated on the basis of a similarity measure, whereas a user may get similar items that share the same or similar features to the items already seen as the same genres, actors, or producers(i.e., in the example of movies). This effect limits the recommendation coverage and limits the diversity of the recommended items. However, in the case of UCF, even when users share similar experiences, they may still have different experiences, which allows them to recommend different/diverse items, hence giving more pleasurable recommendations.

3.2.3 Illustrative example

Let's consider a set of users U which consists of 5 users $U = \{u_1, u_2...u_5\}$ and a set of items I which consists of 8 items $I = \{i_1, i_2...i_8\}$. Table 3.1 demonstrates the ratings expressed by the users in U to the items in I.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	5	4	3	-	-	-	-	-
u_2	4	5	2	5	-	5	-	-
u_3	3	5	1	-	5	2	-	-
u_4	5	-	-	4	-	4	-	-
u_5	-	-	-	2	-	3	4	5

Table 3.1: A simple example of user-item ratings matrix.

Furthermore, we consider Table 3.3 and Table 3.2 which show the correlations weights of each pair of users and items respectively.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0.85	0.35	0.45	0.15	0.75	NAN	NAN
i_2	0.85	1	0.25	0.19	0.20	0.10	NAN	NAN
i_3	0.35	0.25	1	0.28	0.09	0.14	NAN	NAN
i_4	0.45	0.19	0.28	1	NAN	0.47	0.12	0.11
i_5	0.15	0.20	0.09	NAN	1	0.09	NAN	NAN
i_6	0.75	0.10	0.14	0.47	0.09	1	0.22	0.17
i_7	NAN	NAN	NAN	0.12	NAN	0.22	1	0.23
i_8	NAN	NAN	NAN	0.11	NAN	0.17	0.23	1

Table 3.2: A simple example of item-item similarities matrix.

	u_1	u_2	u_3	u_4	u_5
u_1	1	0.70	0.65	0.25	NAN
u_2	0.70	1	0.60	0.45	0.15
u_3	0.65	0.60	1	0.30	0.18
u_4	0.25	0.45	0.30	1	0.17
u_5	NAN	0.15	0.18	0.17	1

Table 3.3: A simple example of user-user similarities matrix.

If we consider the target user u_1 in both cases of memory (user/item)-based collaborative filtering, with the similarity values in previous tables, computed using a measure that considers the number of co-ratings. The recommendation process varies from UBC to IBC as explained in the next elements:

Memory user-based collaborative filtering: The selection of the neighbors to the target user u_1 involves looking at his similarity values to the other users. From Table 1.3, we exclude u_5 which represents a typical case of a sparsity problem where a similarity value can't be computed because of null co-rate items between both users.

For a neighborhood size $k = 2$, it remains three possible neighbors (u_2, u_3, u_4). From Table 1.3, the users u_2 and u_3 can be selected as neighbors, while the user u_4 is excluded even it seems to have similar behavior. This is because it only has one common rating of u_1.

Memory item-based collaborative filtering: Predicting u_1's rating to item i_4 requires looking to similar items rated by u_1. Observing Table 1.1 and Table 1.2 shows that i_4 and i_1 are similar. Thus, we may recommend i_4 because u_1 likes i_1. Additionally, we say that both i_7 and i_8 can't be recommended because they have very few ratings.

3.2.3.1 Features of memory-based collaborative filtering

The use of MeCF can be motivated by its features in comparison to other methods. In [Desrosiers 11], the authors listed several attractive bases, for which MeCF is desirable.

Simplicity: Neighborhood selection is mostly a simple task and depends on a few parameters such as the size of the neighborhood, the minimum closeness threshold, the choice of the similarity measure, and finally a simple prediction formula.

Justifiability: As recommending items is based on selecting close neighbors, these recommendations can be justified simply by the similarity of the users. An example, when an ICF algorithm is applied, a set of similar items that may hold similar genres, are recommended to users, hence an easy way to justify the reason for recommending these items.

Efficiency: Neighborhood selection is an efficient task in comparison to model-based approaches which implies a hard training phase. In addition, the neighbors can be stored efficiently, while more clustering algorithms are developed to scale up to real-world applications that extend to millions of users and items.

Stability: Neighborhood-based approaches are weakly dependable on the change in the user-item rating matrix. In the case of ICF, recommending items to a new user does not require computing any new similarities, while adding more ratings requires re-computing the similarities of only the items that get new ratings.

3.3 Neighborhood selection

The selection of neighbors is a critical step in MeCF algorithm, whereas several techniques were adapted for this objective. In the literature, many works employ various similarity measures, clustering algorithms, and graph theories to explore and select among the wide range of possible neighbors interesting ones.

3.3.1 Unconstrained *k*-nearest neighbor selection

k-nearest neighbor selection (*k*-nn) is a known algorithm in the classification field which was adapted for RS to use in the neighborhood selection. *k*-nn consists of using a similarity measure to create a similarity matrix, whereas for $|U|$ user, a $|U| \cdot (|U| - 1)$ similarity value is calculated. The selection of neighbors is, then, performed by selecting the most k close neighbors. k is a critical parameter that influences the quality of the recommendations [Bobadilla 11a].

3.3.2 Constrained *k*-nearest neighbor selection

On the contrary, to unconstrained *k*-nn, constrained *k*-nn selects all of the neighbors which have a similarity value greater than a minimum threshold. This method has two drawbacks a) it increases the complexity as it requires higher neighbohood sizes. b) it is difficult to fix the minimum threshold value [Mehta 15].

3.3.3 Graph-based neighborhood selection

In graph-based neighborhood approaches, nodes represent users, items, or both of them. The edges between these nodes represent user-user similarities, item-item similarities, or the ratings given by users to items. Graph-based approaches allow propagating information along the edges of the graph between the indirectly connected nodes. Information propagation is greater in the edges of high weights and lowers otherwise. These approaches were used for the recommendation in two ways: a) approximating the similarity between the users or the items. b) approximating the recommendation list by measuring the closeness of an item to a user.

3.3.3.1 Path-based similarity

The distance between the users, the items, or a user and an item can be evaluated in path-based similarity as the number of edges connecting two nodes and the weights of these edges. In the literature, two approaches denoted by the shortest path [Aggarwal 99] and the number of paths [Huang 04] analyze the characteristics of a graph to evaluate the similarity between the users and to generate the recommendations respectively.

3.3.3.2 Random walk-based similarity

Probabilistic-based approaches were incorporated to propagate the information along the edges, whereas the similarity between the users or the items can be evaluated as a probability of reaching these nodes in a random walk [Fouss 07] [Fouss 06].

3.3.4 Evolutionary-based neighborhood selection

Evolutionary algorithms were adopted for exploring the wide range of possible neighbors independently from the similarity measures. In a number of works, genetic algorithms were employed with/without combination to *k*-nn algorithm to select the neighborhood. These techniques employ explicit recommendation quality metrics or neighborhood quality measures to improve the neighborhoods. As an example, in [Karabadji 18], a new encoding was proposed to represent all possible neighborhoods. The proposed encoding holds the neighborhood id, the size of the neighborhood, and the first neighbor id. The search space was explored and expoloited using a fitness function that consists of a linear combination of the diversity and the similarity of the users.

In another example, [Rahman 14] modified *k*-means algorithm such that it can be used together with a genetic algorithm (GA) to deal with the conventional k-means problems such as the initial seeds selection and the number of clusters' parameter. The proposed fitness function tried to minimize the intra-cluster similarity and maximize the inter-cluster distances.

Furthermore, in [Honda 01] and [Wang 14a], a genetic algorithm was used to determine the near-optimal initial seeds by encoding the solutions in the form of a real value vector that holds the possible seeds identifiers. After the optimization, *k*-means algorithm was applied to each survived chromosome solution. Then, the best-resulted chromosome was selected. Moreover, in [Katarya 16], the authors have proposed a clustering technique of three components: *k*-means algorithm, particle swarm optimization (PSO), and fuzzy *c*-means clustering. Firstly, users were categorized according to their preferences towards the items' genres. Then, a combination of PSO and *k*-means algorithm was adopted to select the initial centers which were used to feed c-means algorithm.

3.3.5 Clustering-based neighborhood selection

Clustering algorithms allow identifying within a group of users or items, sub-groups of similar profiles. Clustering algorithms isolate interesting behavioral tendencies, alleviate the cold start problem, and enhance the scalability of the system. The application of *k*-nn algorithm on a clustered set of users, limits the size of candidate neighbors only to the number of members inside the clusters, instead of the whole users' set. Furthermore, cold users are assigned to denser clusters, which improves the quality of the recommendations [Shinde 12].

3.4 Similarity measures

Similarity measures were used to select the set of neighbors of each user and to amount the effect of each neighbor on the rating prediction. The efficacy of similarity measures depends on some factors: a) the level of data sparsity, b) the context of the recommendation and c) the efficiency of the measure.

3.4.1 Correlation-based similarity measures

Measuring the similarity between two users (i.e., or items.) consists of representing the users by sparse profiles (i.e., a set of pairs that includes an item and a given rate or a user and a given rate.). Then, for each pair of users (i.e., or items), co-rated items (i.e., or co-rated users.) must be identified to compute the similarity. Correlation-based similarity calculates the correlation of two profiles and gives a value in the range [-1 to 1]. In the next elements, we present different similarity measures applied only on the users'. However, these similarities are also applicable on the items.

3.4.1.1 Cosine vector similarity

If we consider a user profile U_1 and another user profile U_2, with the same rated items. The cosine vector similarity (CV) [Billsus 98] can be calculated as:

$$CV(u_1, u_2) = \frac{u_1^T u_2}{||u_1||||u_1||} \tag{3.6}$$

3.4.1.2 Pearson correlation similarity

A known formula consists of removing the mean and the variance of the ratings made by the users. In this case, the Pearson correlation [Deshpande 04] is defined as:

$$Pearson(u_1, u_2) = \frac{\sum_{i \in (I_1 \cap I_2)} (R_1^i - \overline{R_1})(R_2^i - \overline{R_2})}{\sqrt{\sum_{i \in (I_1 \cap I_2)} (R_1^i - \overline{R_1})^2 (R_2^i - \overline{R_2})^2}}, \tag{3.7}$$

where R_1^i and R_2^i are the ratings average of the users U_1 and U_2 respectively.

3.4.2 Other similarity measures

3.4.2.1 Mean square difference

Mean square difference (MSD) [Shardanand 95] considers the inverse distance of two users by calculating the cumulative addition of the squared difference of every co-rated item as in the next equation:

$$MSD(u_1, u_2) = \frac{|I_1 \cap I_2|}{\sum_{i \in (I_1 \cap I_2)} (R_i^1 - R_i^2)^2}, \tag{3.8}$$

This measure can be modified to consider the mean or the variance of the ratings, however it still neglects the negative correlation of the users.

3.4.2.2 Spearman rank correlation

Previous measures use the rating values directly and ignore their ranking. Spearman rank correlation (SRC) [Gibbons 90] considers the rank of an item instead and is defined as:

$$SRC(u_1, u_2) = \frac{\sum_{i \in (I_1 \cap I_2)} (k_1^i - \overline{k_1})(k_2^i - \overline{k_2})}{\sqrt{\sum_{i \in (I_1 \cap I_2} (k_1^i - \overline{k_1})^2 \sum_{i \in (I_1 \cap I_2} (k_1^i - \overline{k_1})^2}}, \tag{3.9}$$

where k_1 is the average rank of the items in u_1 profile, k_1^i is rank of item i in u_1's profile.

This similarity has the advantage of avoiding the rating normalization problem.

3.4.3 Similarity based-optimization measures

Optimization algorithms such as PSO, GA were used to directly infer the correlation between two users as done in [Acilar 09], whereas an encoding scheme that holds a matrix of real values was proposed to represent a solution. This method proved a significant enhancement.

3.4.4 Similarity weighting

3.4.4.1 Significance

Similarity significance [Herlocker 02] allows to measure how real two users are close to each other. If we take the example of two users, who have N co-rated item. The simi-

larity of these users can be more significant as N is bigger. It exists many contributions [Bellogín 14], in which, the similarity significance was calculated. A proposition consists of measuring the similarity significance in respect to a minimum co-rated items threshold λ, as in the next:

– Significant weight calculated as:

$$w(u_1, u_2) = \frac{min(|I_1 \cup I_1|, \lambda)}{\lambda}, \tag{3.10}$$

– The similarity between two users is calculated as the product of a conventional similarity measures and the significance weight.

3.4.4.2 Variance

The similarity between the users considers only the ratings given to co-rated items. However, these items differ from each other in terms of the number of ratings which they may get. The items' variance has been integrated [Jin 04] in similarity measures by adapting Pearson correlation coefficient as in the next:

– A weight λ_i is given for each item i as:

$$\lambda_i = log(\frac{|U|}{|U_i|}) \tag{3.11}$$

– Pearson correlation coefficient (or any other measure) can be adapted as:

$$sim(u_1, u_2) = \frac{\sum_{i \in I_{u1} \cap I_{u2}} \lambda_i (r(u,i) - \bar{r_{u1}})(r(u_2,i) - \bar{r_{u2}})}{\sqrt{\sum_{i \in I_{u1} \cap I_{u2}} \lambda_i (r(u,i) - \bar{r_{u1}}) \sum_{i \in I_{u1} \cap I_{u2}} \lambda_i (r(u_2,i) - \bar{r_{u2}})}} \tag{3.12}$$

3.5 Rating prediction

According to [Desrosiers 11], rating prediction can be performed as a regression problem, or a classification problem, whereas the rating type (i.e., continuous values or discrete values) is a key factor on which the suitable rating prediction formula can be chosen.

3.5.1 Rating prediction as a regression problem

As previously cited, neighborhood-based recommendation benefits from the experience of similar users to recommend interesting items. For this end, for each user, u, and

item i not seen by u, a selection of close neighbors set $N_i(u)$, who have seen i according to a similarity measure. Then, an aggregation formula can be performed to approximate the preference of u toward i, which we denote by $\hat{r}(u,i)$. A generic prediction formula can be described as in the next equation.

$$\hat{r}(u,i) = \frac{1}{|N_i(u)|} \sum_{v \in N_i(u)} r(v,i), \tag{3.13}$$

As the neighborhood-based recommendation uses a similarity measure to select neighbors, the closeness of these neighbors must be considered during the rating prediction step. This important note is missed in the previous prediction formula. A possible adaptation consists of weighting v' rating by its similarity to the target user u. The next Equation demonstrates this idea formally.

$$\hat{r}(u,i) = \frac{\sum_{v \in N_i(u)} sim(v,u) \cdot r(v,i)}{\sum_{v \in N_i(u)} sim(v,u)}, \tag{3.14}$$

where, $sim(v,u)$ is the similarity between the users u and v.

3.5.2 Rating prediction as a classification problem

Rating prediction can be computed differently, by considering the votes of each user on a specific rating value. This prediction approach is more appropriate when the rating scale is small, like in the case of binary preferences (e.g., like, dislike), or a small range of preference values like [1-5]. For a user u, and an item i not seen by u, a selection of neighbors $N_i(u|r)$ which has rated i giving a value r, the rank of i can be computed by dividing the size of $N_i(u|r)$ on $N_i(u)$. The next equation demonstrates formally this idea.

$$\hat{r}_{rank}(u,i) = \frac{\sum_{v \in N_i(u|r)} sim(v,u)}{|N_i(u|r)|}, \tag{3.15}$$

Once this rank is computed, the rating $r(u,i)$ is r' value, which corresponds the biggest $\hat{r}_{rank}(u,i)$.

3.5.2.1 Rating normalization

The users express different preferences scales toward the items. In other words, on a rating scale [1-5], a user who is extremely satisfied by an item may give a rating equal to 5. However, another user may give a rating equal to 4 to express the same preference level. This difference affects negatively the precision of the recommendation. In the literature, a common trend to overpass this problem consists of normalizing the ratings. It exists two known normalization methods, which are listed in the next elements:

Mean centering: [Breese 98] [Resnick 94] Statistically, similar users have a close average rating. Hence, computing the difference between a user's ratings and his rating average may show up the positive and negative preferences. This normalization can be computed as in the next:

$$h(r(u,i)) = r(u,i) - \bar{r_u}, \tag{3.16}$$

Z-score normalization: [Herlocker 99] Using mean centering normalization only allows distinguishing positive and negative preferences but not expressing the differences in the satisfaction levels. Giving two users u_1 and u_2, who have the same rating average around 3. User u_1 gives ratings around 3, while u_2 gives various ratings in the range $[1-5]$. In the current example, when a rating equal to 5 given by u_1 expresses very high satisfaction in comparison to the same rating given by u_2. An additional adjustment in z-score normalization allows overpassing this limitation. In the next items, we show how this normalization can be performed in the case of UCF and ICF.

– **User-based collaborative filtering:**

$$h(r(u,i)) = \frac{r(u,i) - \bar{r_u}}{\sigma_u}, \tag{3.17}$$

Equivalently, the rating prediction formula can be adjusted as:

$$\hat{r}_{u,i} = \overline{r_u} + \sigma_u \frac{\sum_{v \in N_u} sim(u,v)(r_{(v,i)} - \overline{r_v})/\sigma_v}{\sum_{v \in N_u} sim(u,v)} \tag{3.18}$$

– **Item-based collaborative filtering:**

$$h(r(u,i)) = \frac{r(u,i) - \bar{r_i}}{\sigma_i}, \tag{3.19}$$

Equivalently, the rating prediction formula can be adjusted as:

$$\hat{r}_{u,i} = \overline{r_i} + \sigma_i \frac{\sum_{j \in N_i} sim(i,j)(r_{(i,j)} - \overline{r_j})/\sigma_j}{\sum_{j \in N_i} sim(i,j)} \tag{3.20}$$

We denote by σ_u,σ_v,σ_i, and σ_j ,in the previous equations, the normalization terms of the users and the items correspondingly.

3.5.3 Rating prediction: classification versus regression

The choice between regression-based rating prediction and classification-based rating prediction can be solved by considering the next factors: the rating scale, the rating

normalization, and the similarities between users. First of all, when the rating values are continuous as in Jester dataset, the regression formula is more suitable for rating prediction, whereas classification-based rating prediction is more suitable for discrete rating values. Secondly, a comparison can be based on the choice of rating normalization method, which transfers the ratings from discreet values to continuous values. Continues values are hard to handle in classification problems. Finally, when users' similarities are close or have small differences, it is more suitable to use a classification method because regression methods may affect the efficiency of the prediction.

3.6 Recommendations ranking

In classical RS, top-N recommendations are recommended to the target user according to their relevancy. The relevancy of the recommendations can easily be explained in MeCF algorithms, as in the case of recommended movies selected based on their similarity to previously seen movies, or movies selected based on the preferences of like-minded users. In other cases, it is hard to justify the selected recommendations like in the case of matrix factorization algorithms, which predict users' ratings using latent preference factors. Recommendation ranking considers a set of candidate recommendations I_c, of $|I_c|$ item associated with their weights/predictions values, and it selects among them the highest weighted items. Algorithm 3.1 explains how items were ranked.

Algorithm 3.1: Rating-based recommendation ranking algorithm.

Data : L_R : *list of recommendations*, L_P : *list of predictions*, N : *size of recommendation list*;
Result : FL_R : *final recommendation list*
while *N!=Size(FL_R) AND !Empty(L_R)* **do**
    i ← getIndexofMaxPrediction(L_p);
    $FL_R \leftarrow FL_R \cup$ getItemWithIndex(L_p,i);
    $L_R \leftarrow L_R \cup$ getItemWithIndex(L_R,i);
    $L_P \leftarrow L_P$ - getItemWithIndex(L_p,i);

3.7 Analysing user-based collaborative filtering

UCF faces several limitations and issues which deserve an analysis to investigate possible enhancements. In the next elements, we list some known issues.

3.7.1 Scalability problem

The scalability issue is caused by the huge number of users and items. It is a known problem in MeCF algorithms. In such a situation, similarity calculation becomes a complex task where filtering algorithms is necessary to save time and resources. The high complexity affects the ability of the system at covering more users and giving them recommendations in a short time. The scalability problem was treated in the literature using: dimensionality reduction, parallel/distributed architectures, and material solutions.

Dimensionality reduction: The statistical-based methods such as PCA, SVD, and their variants were used to extract significant features from the behavioral users' data (i.e., ratings.). This approach restricts the computations on fewer and more significant statistical abstract representations. As an example, applying UCF on more condensed profiles may be of more interest when comparing to the whole set of items' profiles. Moreover, the clustering is a solution that simplifies the data complexity by partitioning the users/items sets into smaller subgroups of similar profiles before performing the recommendations task [Mehta 17] [Zarzour 18] [Symeonidis 16].

Parallel and distributed architectures: This idea consists of using more memory and processors to accelerate the computations. In some real examples, the map-reduce paradigm was proposed to enhance the performance of *k*-nearest neighbors algorithm. Differently, some ideas consist of performing in parallel some recommendation tasks such as the neighborhood selection. However, these solutions are still based-engineering and are still limited face to the problem [Wang 16b] [Hidasi 16] [Godhani 17] [Panigrahi 16] [HewaNadungodage 17].

3.7.2 Long tail problem

In real e-commerce applications, items are distinguishable by the number of ratings they get. Items' ratings number can be explained by two reasons: a) the time of the inclusion in the system, b) and the employed recommendation algorithm (i.e., most CF algorithms ignore newly inserted items.). In fact, cold items (i.e., newly inserted items.) have only few ratings. In such a situation, UCF algorithm tends to recommend the most rated items, while the rest (i.e., less rated) were ignored, which creates a long-tail list of items rarely consumed/selected/rated. Recommending popular items affects the user's satisfaction and decreases the system's revenues [Wang 16a] [Park 08].

3.7.3 Redundancy problem

Another known problem in the recommender system is the similarity of the recommendations. This problem appears especially when recommending items using an ICF algorithm, where the users get bored because of the high similarity of the recommended

items. Additionally, in UCF, the users' tendency toward rare items is limited as we shall present in the next chapters. Thus, the neighborhood selection of a target user may lead to a very limited diversity because of the high similarity of the selected neighbors [Zhang 08] [Zhou 10].

3.7.4 Similarity measures effect

In MeCF algorithms, similarity measures are used to select for each target item or user, a set of similar neighbors. Similarly, the efficiency is affected by the sparsity problem which decreases the ability to catch the closeness (i.e., the correlation.) of two users/items. In fact, two observations can be marked about the users' ratings: a) some users tend to express higher ratings than other users to express the same preference level. This effect was tackled using statistical-based solutions (i.e., variance, mean, biases, ..etc.). b) some users have only few ratings (i.e., cold users.). Similarity significance weights were used to assign more importance to users having more reliability (i.e., the ones who have a higher number of co-rated items.).

3.8 Recommendation diversification

Recommendation diversification took more interest due to the increasing awareness about the importance of novelty and diversity in enhancing the users' experiences. Many techniques were proposed in the literature to target these objectives and were divided into many categories. In more detail, two main diversification techniques denoted recommendation re-ranking, and learn to rank are known in the literature. However, we add in particular, the clustering algorithms and the evolutionary algorithms as additional categories, which collect many contributions. In the next subsections, we explain these methods in more detail.

3.8.1 Recommendation re-ranking

Re-ranking approaches apply a secondary ranking step before delivering the final recommendation lists as illustrated in Figure 3.1. Firstly, the recommendation accuracy is targeted using a baseline recommender system. Secondly, the recommendations lists are re-ranked using a greedy ranking algorithm. In this step, the recommendations are selected incrementally according to two criteria: a) their relevance to the users' preferences, and b) their contribution to diversifying the recommendation list.

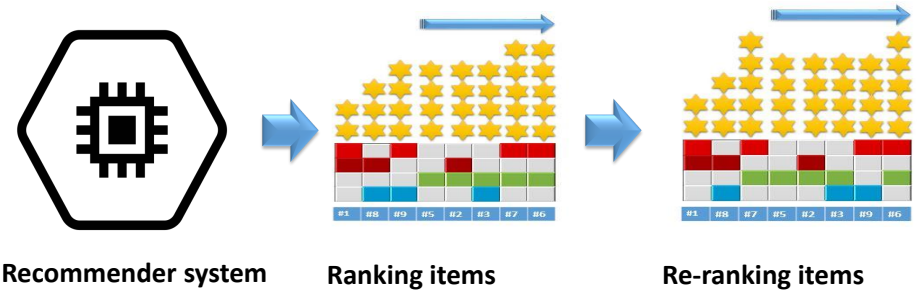


Figure 3.1: Recommendation diversification steps using re-ranking techniques.

The trade-off between these two criteria is achieved by the use of a linear ranking function (i.e., $i^* = argmax_{i \in R-S}(1-\lambda) \cdot rel(i,S) + \lambda \cdot div(i,S)$, whereas λ is a controlling parameter, R is the recommendation set, S is the current selected items, $rel(i,S)$ is the relevancy of i in respect to S and $div(i,S)$ is the diversity which item i brings to S). The maximal marginal relevance (MMR) [Carbonell 98] is originally an information retrieval technique, which was applied for ranking problems in RS field. MMR originally selects incrementally a document, which has the maximum similarity with the user's query and also a minimum similarity to the previously selected items.

Furthermore, xQuAD (eXplicit Query Aspect Diversification) [Santos 10] is a probabilistic model that uses MMR for document diversification by introducing the document relevancy probability toward a query and the aspect diversity probability. This diversity is assured by generating from the main query subqueries each of which has an important weight.

xQuAD was adapted in [Kaya 18], to identify within the same profile different users' tastes (aspects). The extracted profiles' aspects were used in the re-ranking algorithm as an alternative to the explicit features which were employed in the conventional intent-aware diversification recommenders. Another probabilistic model was proposed in [Vargas 14], to improve the diversity by formally representing the genres' probabilities in a linear combination of their global and local appearance probability over the items set and the users' profiles. The probabilistic model defines the genres' coverage and the genres' no-redundancy as two different concepts to use within a diversification greedy re-ranking algorithm. Furthermore, the abased graph re-ranking technique was proposed in [Parambath 16], as a competitor of greedy re-ranking algorithms, whereas the Frank-Wolfe algorithm was employed to ensure a balance between the recommendations' accuracy and coverage.

Figure 3.2: Recommendation diversification steps using learning to rank techniques.

3.8.2 Learn to to rank

The conventional recommendation techniques target only the relevancy, while more important factors such as novelty and diversity were neglected. Thus, recent algorithms were found to balance these recommendation qualities by learning to rank techniques illustrated in Figure 3.2.

In MeCF, new similarity measures were adjusted to promote novelty and diversity. In [Chatzicharalampous 15], the authors presented an adjusted UCF that enhances diversity, whereas a new measure was introduced to select, during the neighborhood selection, more explorer users. An explorer user tends to interact more with rare items, which leads to alleviating the LT problem and improving diversity. Furthermore, in [Wu 16], the authors analyzed the neighborhood selection in terms of the relevancy and the coverage from two different aspects. On one hand, considering the users' interest (i.e., the ratings.) allows selecting similar neighbors to the target user. Additionally, considering the number of unpopular items in the neighborhood set allows measuring the recommendation coverage. On the other hand, the recommendation relevancy was calculated by averaging the prediction values of the items recommended by the neighborhood, while the neighbors' coverage was calculated by counting how many times the users participate in forming the final recommendation set.

Moreover, matrix factorization algorithms (MF) [Bokde 15] address the accuracy by estimating the users' preferences matrix R from the product of two sub-matrices (i.e., $\hat{R} = PQ$, P and Q represent the users and the items in form of latent factor vectors.). Related contributions to MF, differ in terms of the employed data, the definition of the objective function, and the objective of the recommendation model. MF approaches have firstly targeted minimizing the rating prediction error and were later adjusted to fit the Top-N recommendation [Ning 11] [Christakopoulou 16]. MF was

used for the diversification purpose by mainly adapting the objective function to fit this purpose[Gogna 17][Abdollahpouri 17][Wasilewski 16] [Coba 18]. As an example, [Gogna 17] balances between the recommendation accuracy and the diversity by minimizing the variance of the latent factors since lower variance leads to more distributed preferences over the set of items.

3.8.3 Clustering

The clustering techniques were considered for MeCF algorithms to improve the recommendation accuracy. In [Dakhel 11][Sarwar 02b], the authors mentioned the clustering problems related to data sparsity and scalability of the system, where *k*-means clustering was employed in order to split the users set into groups of similar profiles. Also, a scalable co-clustering approach [Wu 16] was presented, in which the authors argue that a given user behaves differently according to different topics. For this reason, both the set of items and the set of users were grouped into groups of similar members which allows applying CF on each subgroup separately. The final recommendations were, then, ranked following an aggregation function. Similarly, the same approach was proposed to tackle the cold-start problem by initially extending the users/items profiles [Pereira 15]. The users were later assigned randomly to different clusters. Finally, to optimize the clustering quality, a prediction error based on a regression formulation was applied.

From a different perspective, items clustering was used in [Aytekin 13] to improve items diversity by assuming a metric that indicates for each user the number of items that s/he needs to get from eachcluster. Additionally, items clustering was applied in [Boim 11], for re-ranking purposes, whereas the clusters were defined by their medoids and the users were assigned to each cluster according to a predefined priority-medoid concept which consists of the next two criteria: 1) Each item should have a rating lower than its representative. 2) Each item should belong to the closest medoid. The importance of the proposition is to avoid putting together similar items while keeping a good accuracy inside each new-formed cluster.

In [Raja 18] a probabilistic matrix factorization model (PMF) was applied on ratings' timestamps, users profiles, and items features to represent the users/items into latent factors vectors. Then, *k*-means algorithm was applied in combination with PSO to cluster the users. In every cluster, expert users were identified with respect to their closeness to other users in terms of the preferred features. Features values taken from expert users were integrated into profiles of users from the same cluster, to enrich their preferences lists. Finally, the recommendations were generated according to the users' preferences toward the features of the movies, and then re-ranked to promote novelty.

3.8.4 Evolutionary algorithms

Bio-inspired algorithms are known for their efficiency in optimizing hard search problems. These approaches were used over RS' state of the art for many objectives such as targeting the selection of the initial seed in *k*-means using genetic algorithms [Kim 08], artificial immune network [Acilar 09], and artificial bee colony [Ju 13]. However, more works have targeted the quality of the recommendation in terms of diversity.

In effect, GA was proposed to enhance the aggregate diversity by firstly reducing the users' set search space using *k*-means, then ranking the recommendations set using GA algorithm with the aim of balancing the accuracy and the diversity [Zuo 15]. In more detail, each cluster was encoded into two dimensions matrix, a first dimension represents the cluster members while the second one represents the users' recommendations. Furthermore, GA was used to optimize hybridization weights of many recommenders with the aim of maximizing the accuracy, the novelty, and the diversity [Ribeiro 14]. Additionally, the ICF algorithm was used for the recommendation in combination with a GA to address the accuracy-diversity trade-off [Wang 14b]. The authors have used a similar encoding to [Zuo 15] and employed two competitive fitness functions to refine the recommendations set. Moreover, GA encoding was developed to target the neighborhood selection problem [Karabadji 18], whereas the chromosomes were encoded to hold three genres, the size of the neighborhood, the identifier of the smallest set of the neighborhood, and the identifier of the neighborhood. The solutions were optimized by a fitness function that combines the users' diversity and relevancy.

Furthermore, In [Bag 19] a modified similarity measure was proposed, to select for each user the most K significant nearest neighbors (SNN) with the aim of maximizing relevancy and diversity. Then, a neural network was used to optimize the similarity weights of each neighbor. A review of the state-of-the-art demonstrated limited use of clustering to improve diversity. In previous contributions, [Berbague 18b][Chatzicharalampous 15], adjusting similarity measures has proven a beneficial effect on enhancing novelty and diversity. These measures care only for the diversity/similarity of every pair of users and neglect the diversity of the whole neighborhood. A different approach [Bag 19] [Karabadji 18] aims to optimize the quality of the neighborhood members of each user individually, resulting in a better quality of recommendations. However, this solution implies an optimization problem for each user. In [Zhang 09], a clustering algorithm was proposed for diversifying the recommendation lists. In their research, each user profile was partitioned, then a set of candidate items was selected with respect to their proximity to the target user. Finally, a ranking method was adopted to advance the itemsâ diversity. In [CAI 20], the authors proposed a hybrid recommender system using an evolutionary algorithms. In their proposition, different basic recommendation algorihms were mixed to improve the recommendation diversity. In tourism applications [Logesh 19b], many PSO algorithms were used to cluster the users, and consequently recommending diversified point of interests.

3.9 Conclusion

MeCF algorithm can be characterized by its simple deployment concept, which consists of selecting a neighborhood for each user, then predicting the preferences of the target user before ranking them. However, they suffer two major problems which were addressed in the state-of-the-art. Firstly, the scalability problem limits the ability of the system at picking relevant recommendations in a tolerable time, whereas an increasing number of users is concerned. Secondly, the recommendation quality is limited in terms of diversity and relevancy because of the data sparsity, the limitations of the similarity measures, the recommendation ranking, and the neighborhood selection methods. On the contrary to the conventional recommenders which adopt statistical-based techniques, we believe that evolutionary algorithms are more encouraging solutions that allow to optimize many recommendation settings at once and to address more than one recommendation quality. In this thesis, we try to investigate new methods to solve these issues. In particular, we give interest to evolutionary algorithms by proposing new schemes and objective functions which we believe in their ability at giving better performances.

The computing scientist ′ s main challenge is not to get confused by the complexities of his own making.

Edsger W. Dijkstra

4

Using genetic algorithms to enhance memory-based collaborative filtering

▷ *This chapter presents genetic algorithms and a detailed introduction to their theories, and working mechanisms. Additionally, we present our propositions to alleviate related problems in the RS field, as in particular the scalability problem and the recommendation quality. In more detail, we criticized k-means algorithm as a known data reduction technique and proposed a genetic-based clustering algorithm instead. Later, we investigated different possibilities to use clustering for multi-objective optimization and analyzed the similarity measures to address the same objective.* ◁

Chapter outline

In UCF algorithms, the recommendation quality strongly depends on the neighbors' selection which is a high computation complexity task in large-scale datasets. A common approach to overpass this limitation consists of clustering the users into groups of similar profiles and restrict the neighbors' selection into the cluster that includes the target user. Genetic algorithms can be used efficiently to solve different problems by proposing adequate encodings. In this chapter, we evaluate the clustering algorithms from the state of the art and the use of genetic algorithms for various objectives. Then, we validate our propositions using different quality metrics and show their superiority against baseline techniques.

4.1 Introduction

RS is an extension of conventional information retrieval techniques. It plays a major role in e-commerce websites as well as social networks such as Google, Facebook, Netflix, and others. RS aims to estimate the users' ratings or to personalize a bag of recommendations to every user based on his preferences made explicitly by rating or implicitly during the interaction with the system (i.e., browsing sequences.). Besides, contextual information (i.e., time, location.), as well as social and demographic information can be used to make high-quality recommendations. In fact, it exists a variety of recommendation algorithms each of which fits a specific type of information and deals with a well-defined recommendation limitation.

Firstly, content-based filtering (CBF) algorithms [Alhijawi 16b] use the available information about the items (i.e., features, attributes.) to make recommendations for a target user. It looks for a set of nearest neighbors among the items already seen by the target user to estimate his preference. CBF is a very beneficial filtering algorithm that works well with the cold start problem (i.e., low users overlap.) by focusing only the users' profile. However, it involves a hard content preprocessing step that raises a complicated task especially while extracting features from media content (i.e., images, videos.). Besides, for a given user, CBF algorithm tends to recommend a set of very similar items to those already seen, by consequence limiting the diversity of recommendations. This effect is denoted in the literature by the overspecialization problem.

Secondly, memory-based collaborative filtering (MeCF) algorithms [Ribeiro 12, Karabadji 18] have given good results in terms of accuracy and were used widely over different researches [Guimarães 13]. UCF bases its recommendation on the concept of the collective users' trend toward the items. This algorithm predicts unrated items of a specific user by considering the similarities with the rest of the users, so close users can strongly influence the predicted rating value while far users would have a limited effect. The similarity distance between the users may be computed using different similarity measures such as cosine similarity, Pearson correlation similarity..etc. *K*-nearest neighbors (*k*-NN) [Ribeiro 12] is the most used algorithm for collaborative filtering.

k-NN takes into consideration only a limited number of users. It determines for a given user *k* nearest neighbors, then calculates the rating prediction for the unrated items by aggregating the ratings of the nearest users. In some contributions [Karabadji 18], the size of neighborhood selection and the minimum similarity between the neighbors were restricted.

An ideal collaborative filtering engine should cope with the scalability problem, and the recommendation quality at once. In fact, the rating prediction, over the whole number of users and items, might be a very complicated task and may involve a long high time complexity. A common approach to deal with scalability issues is to make a preprocessing step that consists of clustering the users set into groups of similar profiles and restrict the collaborative computing on the scale of the cluster to which the target user belongs.

In addition to the scalability problem, the recommendation quality is another research subject that involves many investigations to improve it. As previously explained, recommendation quality has a binomial effect on the system and the users at once. In this chapter, we present different works in which evolutionary algorithms were used to deal with the scalability problem. Later, we investigate the possible techniques to improve the recommendation diversity in UCF. In the next elements, we summarize the main contributions of this chapter:

- We propose a fully based-genetic clustering algorithm to target the scalability problem.
- We incorporate an adapted similarity measure within a two-stage diversification approach, where a post-filtering algorithm based on the use of a genetic algorithm was performed.
- We extend the previous proposition to a more sophisticated clustering algorithm and present further details and comparisons.

In the next sections, we define genetic algorithms before delving deeper into the presentation of each of these elements, with their details.

4.2 Genetic algorithms

We proposed the use of genetic algorithms to deal with users clustering for the next reasons:

- Users' preferences information is insufficient, whereas the users only express few ratings while interacting with the recommendations system. Thus, it is hard to achieve good precision. Genetic algorithms allow the integration of more data

sources, to combine many recommender systems, and to improve the similarity measures.

- Recommendation quality can be driven by real indicators during optimization, whereas real quality indicators such as MAE or Precision were incorporated in the fitness function which may improve the recommendation quality even when users' data are sparse since the optimization, in this case, is based on the recommendation quality and not the quality of the neighborhood, neither the similarity measure.
- GA allows a better, and faster exploration of the search space, whereas the quality of a neighborhood is evaluated as a whole in terms of more than one quality metric (i.e, such as the intra-class similarity.). However, in traditional methods, a neighborhood can be composed only by evaluating the pairwise similarity between each pair of users.
- GA allows handling more than one problem at once, as we can describe exactly the desired recommendation quality by (a) encoding the solutions, (b) fixing the fitness function, and (c) identifying the parameters which we need to integrate.

4.2.1 Definition

Genetic algorithms are machine learning tools for optimization, which are inspired by nature. The idea consists of exploring a large search space to select among the possible solutions, the most suitable/fit solution. Initially, a set of chromosomes/solutions are created with respect to a predefined problem' representation. Chromosomes can specify the required settings of the problem as a suite of genes. The initial set of solutions is mostly created randomly to ensure diversity. Then, for a number of iterations, the initial solutions set evolves to improve its quality by performing three genetic operators: mutation, crossover, and selection. The evolution step is guided by a fitness function that includes one or more objectives. Among the last solutions, the best one can be chosen. In Figure 4.1, an illustration of the genetic-based optimization process.

Figure 4.1: General scheme of genetic-based optimization process.

4.2.2 Genetic operators

Crossover: This operator consists of creating new children from two parents by swapping randomly different fragments of the parents, whereas each fragment consists of a set of successive genes. The crossover operator differs in terms of the number of fragments involved during the operation.

Mutation: This operator is applied to each chromosome independently from the others, and it consists of randomly changing the value of a gene or more to different values.

Selection: this operator allows passing from the previous generation to the next one, whereas for a given set of solutions a portion of it is firstly selected according to a predefined probability distribution that considers the fitness of each encoded solution. This selection can be then subject to mutation and crossover operations. Some selection methods consist of picking the most suitable solutions(i.e., having the best fitness values). Other methods seek to select portions of both good and bad solutions. Another

type of method performs a random selection. The resulted individuals of mutation and crossover operations are denoted by $offspring$ set. Among the initial solutions set and the $offspring$ set, another selection operation can be performed to identify the new members of the next generation.

4.3 Genetic-based clustering algorithm to improve scalability (GA-CLUS)

k-means is a partitioning algorithm used widely in recommendation systems to address the scalability issue. The algorithm is known for its suitability for large-scale datasets and fast convergence, however it is limited by the local optimality solutions. The initialization step plays a crucial role in clustering quality. Random-based initialization of *k*-means in the classical version influences strongly the quality of the obtained results. *k*-means clustering involves specifying initially a fixed number of clusters, the algorithm stops after getting a lower error rate according to a predefined threshold or after reaching a number of iteration.

In more detail, the algorithm assigns each data vector to the closest center and iteratively looks for the best center inside each newly formed cluster. Clustering results depend on the initially chosen seeds which raises the worst drawback.

Many papers have addressed the initial seeds selection dilemma in *k*-means algorithm where two considerable solutions were proposed. The first one is based on statistical basics while the second one is based on evolutionary algorithms. The first type scans data samples to look for their statistical features such as the density and the variance. These measurements were used to identify the appropriate initial seeds. By revenge, evolutionary approaches explore the search space of the possible solutions by formulating the clustering problem and optimizing a clustering quality measure. As an example, the clusters 'density and the recommendation mean square error (MAE) were used to guide the optimization.

In this contribution, we present a partitioning-based genetic algorithm (GA-CLUS) to enhance UCF. This objective is achieved by pulling up an optimal partitioning of users over *k* groups. This may allow us to deal with the scalability problem in recommender systems. The proposed optimization algorithm guided by a clustering quality measure explores possible solutions over the whole possibilities allowing for partitioning a set of n users into k non-empty subsets. A summary of our contribution can be listed in the next elements: a) proposing a new genetic encoding to represent the possible solutions by a set of centers and the number of the most similar users around them. b) designing a multi-objective optimization fitness function to ensure the intra-group similarity and diversity between centers.

4.3.1 Clustering encoding scheme of GA-CLUS

GA-CLUS encoding consists of representing the solutions as arrays of 0s and 1s (i.e., binary strings). Therefore, we propose a new binary string encoding, each one must store the full information about its corresponding partitioning solution. Our designed encoding is presented in Figure 4.2, where a chromosome is composed of k genes and each gene stores two information: a) an integer C_i representing a center; b) an integer α_i representing the number of most nearest users over the center C_i. This encoding allows

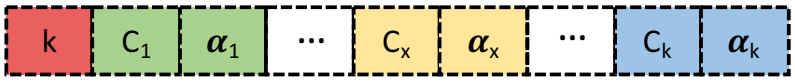


Figure 4.2: Chromosome encoding scheme of GA-CLUS algorithm.

representing the whole possible partitioning configurations, where k is the number of partitions for which we take as parameters two thresholds minimum Min_C and maximum Max_C number of clusters and look for the best users partition over k groups. Each group (i.e., cluster) of the k ones is represented by its center C_i and the number of its members α_i. A C_i storage space is represented by X_C bits which encode a user identifier $[1..|U|]$. Even here we can note that the C_i identifiers must be different. While α_i storage space is represented by X_α bits which encode members in each group and their sum must be equal to $(|U| - k)$. The chromosome size $|ch|$ is number of bits need to represent in binary: 1) Max_C; 2) $|U|$; 3) $|U| - k$. Thus $|ch| = X_{Max_C} + (Max_C * (X_\alpha + X_\alpha))$. To put it simply we propose to see the next Example.

Example Assume a data with $|U| = 200$. Thus, the minimum Min_C and the maximum Max_C number of clusters are 2 and 10 respectively if the size of the cluster including the center is at least 20. Therefore, to encode all possible configurations, we consider the largest case which requires the greatest storage space. For this instance case we need to encode information over $X_k + Max_C * (X_C + X_\alpha)$ bits= 164 bits. First, to encode the number of clusters, 4 bits are required to store a binary representation of integers between 2 and 10. Then, a maximum size value of a group including a center may be 180 for two clusters partition. According to this maximum size value (i.e., 180), the storage space, X_α required is 8 bits. While the most value of center identifier maybe 200, then the storage space X_C required is 8 bits.

4.3.2 Clustering scheme decoding of GA-CLUS

The decoding steps allow converting the binary gene codes encoded in a chromosome ch into the appropriate k groups. This conversion consists of determining centers' identifiers (i.e., users considered as centers.) and the groups' size over these centers. Mainly this decoding phase follows two steps: a) Binary subsequence represented the group's

number is converted to the appropriate integer (i.e., k). b) According to this group's number, couples of binary subsequences represented centers and groups size are converted to the appropriate user identifier and group member size (i.e., an integer).

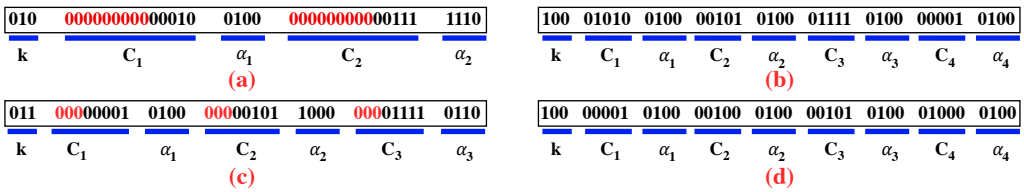


Figure 4.3: Chromosome instances of GA-CLUS algorithm.

Assume a data with $|U| = 20$, a minimum Min_C and a maximum Max_C number of clusters equal to 2 and 4 respectively. The search space is composed of the whole partition ways which are equivalent to the ways of writing $|U|$ as a sum of positive integers by considering all possible permutations. Figure 4.3 illustrates 4 chromosome instances that encode 4 different solutions which could be decoded into:(a) $k = 2$, ($C_1 = 2$, $\alpha_1 = 4$), ($C_2 = 5$, $\alpha_2 = 14$);(b) $k = 4$, ($C_1 = 10$, $\alpha_1 = 4$), ($C_2 = 5$, $\alpha_2 = 4$), ($C_3 = 15$, $\alpha_3 = 4$), ($C_4 = 1$, $\alpha_4 = 4$);(c) $k = 3$, ($C_1 = 1$, $\alpha_1 = 4$), ($C_2 = 5$, $\alpha_2 = 8$), ($C_3 = 15$, $\alpha_3 = 6$);(d) $k = 4$, ($C_1 = 1$, $\alpha_1 = 4$), ($C_2 = 4$, $\alpha_2 = 4$), ($C_3 = 5$, $\alpha_3 = 4$),($C_4 = 8$, $\alpha_4 = 4$).

All chromosomes are 0's and 1's strings of a length equals 39. The clusters number is encoded in 3 bits. Cluster centers C_i are encoded at least on 5 bits and at most 9 bits (i.e., 5 bits for chromosomes (b) and (d), 6 bits for chromosome (c), and 9 bits for chromosome (a)). Group size is encoded in 4 bits. According to the decoding phase: first, the k clusters number is designed where the three first bits of chromosomes are converted to integer (i.e., 010⇔2, 100⇔4, 011⇔3, 100⇔4 for chromosomes (a), (b), (c) and (d) respectively). Next, using k we can determine the centers' identifiers and the size of each group. The number of bits required to encode centers is $log_2(|U|)$. However, if k is less than Max_C, extra 0's bits are added to X_c. According to this example, nine 0's and three 0's bits are added to centers identifier bits on chromosomes (a) and (b) respectively. Converting these centers' identifiers bits allows identifying the corresponding users. Then, bits representing groups' sizes are converted to a set of integer values. Finally, each chromosome is converted to its correspondent groups, where each center C_i and it is α_i nearest neighborhoods compose a cluster.

4.3.3 Clustering quality optimization of GA-CLUS

A good clustering quality in the recommendation context implies maximizing accuracy on the level of each cluster as well as keeping a meaningful distance between centers. Our fitness function combines these two measures. Thus, we define our fitness function as the next:

$$group\_precision(ch) = (max(r) - min(r)) - (\frac{1}{k} \times (\sum_{i=1}^{k} MAE(G_i)) \tag{4.1}$$

$$center\_diversity(ch) = \frac{1}{k \times (k-1)} \times (\sum_{i=1}^{k} \sum_{j=i+1}^{k-1} (1 - sim(C_i, C_{j+1}))) \tag{4.2}$$

$$fitness(ch) = group\_precision(ch) + center\_diversity(ch) \tag{4.3}$$

where, $max(r)$ and $min(r)$ denote the biggest and lowest rating values equivalent to 5 and 1 respectively, G_i is the i_th cluster and C_i is the center of the cluster i. In fact, we have chosen to combine two statistical terms: the first one has the goal of giving a significant indicator of centers' positions. Logically, that two centers should not be very close or belonging to the same cluster. However, maximizing the distances between the pairs of centers push the GA to search for the set of the farthest users in the search space. Thus, we added a second term to control the internal prediction error in each cluster.

4.3.4 Experimental scenario

In this section, we validate the proposed clustering algorithm experimentally. The obtained results were compared to both *k*-NN algorithm, in addition to *k*-means and *PCA-GAKM* which are well-known data reduction algorithms. We performed the experiments on Movielens 100k described in Chapter 2. We split the dataset into 90% for training and 10% for testing. Furthermore, we evaluated the results using: MAE, Recall, and Precision metrics.

The configuration of the proposed GA consists of initializing the number of the chromosomes at 20, the maximal iteration number at 200, while for both *k*-means and *PCA-GAKM*, we chose the best clustering configuration that minimizes their MAE.

4.3.5 Analyzing neighborhood size parameter

CF algorithm involves two stages:(a) the neighborhood selection process, and (b) the rating prediction. In particular, the neighborhood selection deals with two important parameters expressly the distance measure choice and the size of the neighborhood. For the first parameter, we employed the Euclidean distance similarity. However, for the second one, we increased the size of the neighborhood from 5 to 60 by an increment of 5 each time.

k-NN algorithm was applied one time on the whole dataset and again on each cluster separately for the different clustering algorithms: GA-based clustering, *k*-means, and PCA-GAKM. In more detail, we applied different evaluation metrics on *k*-NN considering both rating accuracy (i.e. using MAE) and recommendation set evaluators

(i.e., precision and recall.). For all evaluations, we set the size of the recommendation list equal to 5.

Rating prediction accuracy

Figures 4.4 and 4.5 demonstrate the superiority of our algorithm against both k-NN and the clustering techniques. We observe that MAE trace descends when increasing the size of the neighborhood. In fact, the performance of k-means clustering is almost stable and ranges between 0.81 and 0.82, while our algorithm gives widely better results than k-NN. Again, our algorithm gives better results than $PCA - GAKM$ when the neighborhood size is upper than 40.

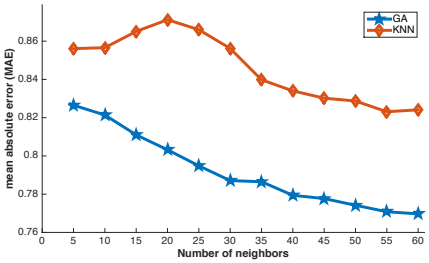


Figure 4.4: Comparison of GA-CLUS to k-NN in term of MAE.

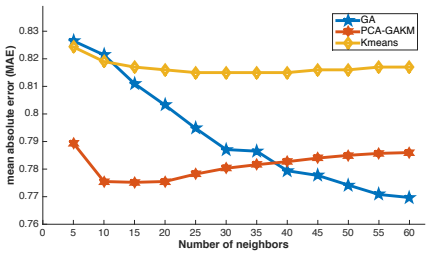


Figure 4.5: Comparison of GA-CLUS to k-means and PCA-GAKM in term of MAE

4.3.5.1 Recommendation set accuracy

We examined our proposed algorithm in terms of Precision and Recall against k-NN and k-means algorithm. Figures 4.6 and 4.7 show the superiority of the clustering algorithms to k-NN, where our proposed algorithm gives better perfomance than k-means when the neighborhood size passes 30.

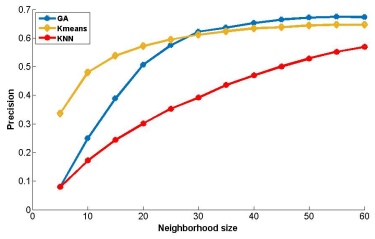


Figure 4.6: Comparison of GA-CLUS to other algorithms in term of Precision.

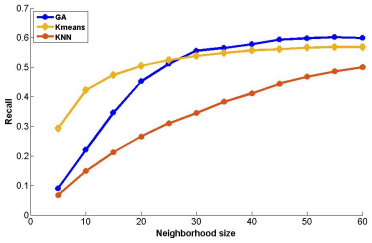


Figure 4.7: Comparison of GA-CLUS to other algorithms in term of Recall.

4.3.6 Analyzing the recommendation length parameter

We analyze, in this section, the length of the recommendation in terms of Accuracy and Recall. Thus, we increased the number of the recommendation from 1 to 20. Figures 4.8 and 4.9 show Precision and Recall variation while increasing the recommendation length. We observe that Precision corresponds to the recommendation length inversely while Recall corresponds to it directly. Additionally, both clustering algorithms keep giving better results than *k*-NN with a clear superiority of our GA against *k*-means.

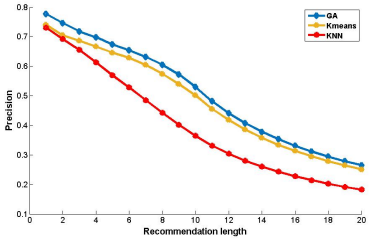


Figure 4.8: Comparison of GA-CLUS using different recommendation lengths in term of Precision

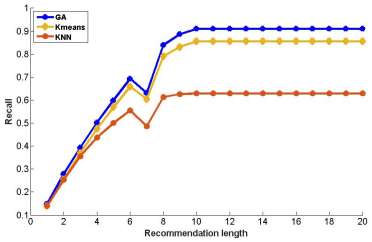


Figure 4.9: Comparison of GA-CLUS using different recommendation length in term of Recall

As a summary, ranging *k* between 5 and 60 has shown the superiority of clustering algorithms to *k*-NN (i.e., which is the most complex algorithm.) in terms of Precision. Also, results show a positive correlation between the high neighborhood sizes, the recommendation accuracy, and the negative effect on the scalability. This can be explained by the openness of the neighborhood selection to include the whole set of users in the case of *k*-NN, while it is limited in the case of clustering algorithms on the level of the cluster to which the target user belongs. In effect, the clustering approach has achieved a better balance between the accuracy and the scalability with the superiority of our proposed algorithm to *k*-NN, *k*-means, and PCA-GAKM algorithms.

4.3.7 Discussion

The scalability problem is a major drawback in MeCF algorithm. We have targeted this issue by proposing a genetic-based partitioning algorithm with the aim of better clustering the users. The treated problem was encoded to reduce the search space by indicating the number of possible clusters as well as specifying their maximum and minimum size. Besides that, we maximized the quality of clustering in a way to achieve more accurate recommendations. However, the sparsity problem might influence the similarity performance between the pairs of users. Many statistical techniques were used to solve this problem. In addition, the importance of the recommendation accuracy is insufficient to evaluate the satisfaction of the users. Pushed by the recommendation re-

dundancy, the diversity and the novelty of the recommendations earned more interest. Our next intent is to adapt the clustering for the recommendation diversification.

4.4 A two-stage improved k-NN algorithm to enhance diversity (TS-Ik-NN)

In this section, we verify incorporating the users' pairwise diversity within a similarity measure to investigate the possibility of getting richer neighborhoods in terms of diversity and novelty. This idea was discussed in [Berbague 18a], where we showed a two-stage neighborhood selection approach. In a first step, we sought to reduce the search space extension by performing an adapted k-NN algorithm. In this stage, we modified a similarity measure to combine a pairwise user diversity measure and a similarity-based rating measure. Then, in a second step, we employed a genetic algorithm to improve the neighborhood selection. During this step, we explored for each user the possible neighborhood sets obtained from the first step to select among them the best ones which may improve the recommendation quality. The objective of the proposition is to enhance the scalability of the system and to improve the recommendation quality.

4.4.1 First step: novelty-based neighborhood selection

Using conventional similarity measures allows selecting similar neighbors and consequently limiting the diversity of the recommendations. However, considering the diversity within a similarity measure may allow obtaining a dual control on the users set to select similar and diverse ones at the same time. Following this conception, we propose the use of the next similarity formula:

$$new\_sim(u_1, u_2) = \alpha \times sim(u_1, u_2) + (1 - \alpha) \times div(u_1, u_2), \tag{4.4}$$

where, α is a controlling parameter, and $div(u_1, u_2)$ is a diversity measure between the users u_1 and u_2.

For the similarity calculation, many similarity measures are known in the state-of-the-art. However, the diversity calculation involves new metrics. The quality of the recommendation is a result of the whole contributions from every member of the neighborhood. In the aim of selecting neighbors with potential diversification abilities, we propose the next diversity measure:

Popularity: the users who can decrease the popularity effect are the ones who tend to select rare items from the market. Thus, a good measure of the user pref-

erence toward the less popular items could be the next:

$$div(u_1, u_2) = \sum_{i \in I_2 - I_1} 1 - \frac{P(i)}{|U|}, \tag{4.5}$$

where, $P(i)$ is the number of the users who rated item i.

The adjusted similarity measure allows only one to one selection (i.e., target user and candidate neighbor.) and neglects the similarity of the neighborhood set as a whole. Therefore, we propose the use of a genetic algorithm as a complementary step that allows optimizing the quality of the neighborhood.

4.4.2 Second step: using genetic algorithm to optimize the quality of the neighborhoods

The objective of the proposed scheme is to address two well-known issues in the recommendation field: the scalability of the system and the recommendation diversity. In fact, each user gets a set of possible neighbors using an adapted similarity measure. The assignment of the neighbors allows for each user to have a view over the whole set of users and to increase the neighbors' diversity and similarity.

Initialization: the importance of the initialization step depends on the complexity and the nature of the problem. We adopted a random chromosome generation. Also, we set as parameters the number of chromosomes equal to 20, the mutation probability equal to 0.2, and the crossover probability equal to 0.7. Finally, we adopted a binary tournament selection strategy.

Fitness function: the fitness function choice has a crucial role in accelerating the approximation as well as improving the desired recommendation quality. We used, in our case, a linear combination of Precision and Diversity as a function to guide the exploration process.

The balance between the recommendation relevancy and the recommendation diversity is controlled during the evolutionary optimization using the next fitness function:

$$fitness(ch) = \beta \times (1 - Precision) + (1 - \beta) \times (1 - Diversity), \tag{4.6}$$

where, β is a controlling parameter.

Encoding: the chromosome encoding, as illustrated in Figure 4.10, holds for each user his possible neighbors. Thus, each user occupies a length of K binary bits. The K bits represent the candidate neighbors existing in the selected neighborhood set. A bit equals to one means that the neighbor could participate in the recommendation generation, while a value of zero means that the neighbor is excluded.

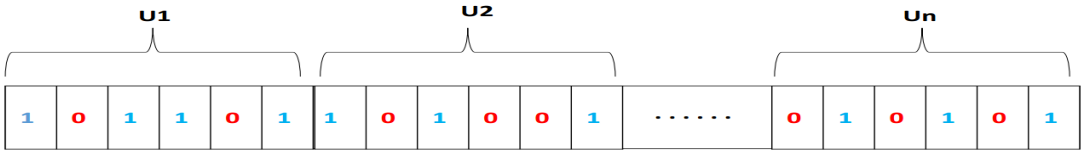


Figure 4.10: Chromosome encoding scheme of TS-I*k*-NN algorithm.

Decoding: the main advantage of the encoding which we proposed is its easy and fast interpretation. The chromosome encoding scheme is used to approve or exclude the candidate users from the neighborhood set.

We have adopted during the similarity calculation an additional criterion. Thus, a user can get recommendations from a possible neighbor only if this neighbor can also get recommendations from the current user. Algorithm 4.1 explains the similarity calculation using our GA.

Algorithm 4.1: Similarity calculation algorithm in TS-I*k*-NN.

Data : weights, U_1, U_2, indexer, neighborhood, similarity;
Result : weighted similarity
if $U_2 \notin$ *neighborhood*(U_1) *OR* $U_1 \notin$ *neighborhood*(U_2) **then**
    return 0;
else
    i $\leftarrow$ indexer(U_1, U_2);
    j $\leftarrow$ indexer(U_2, U_1);
    if *weights(i) =1 AND weights(j) =1* **then**
        return similarity(U_1, U_2);
    else
        return 0;

4.4.3 Experimental design and results comparison

We applied our algorithm on movielens 100k. The results were compared to two baseline recommender algorithms: *k*-NN algorithm and *k*-means algorithm. Results were obtained by changing the neighborhood size in a range of values between 10 and 50 by incrementally adding 10 in every test. Furthermore, we compared against LDA [Xie 15] recommender, which is a model-based recommendation reference. For each recommender system, we choose the best results for accuracy and diversity. The desired recommendation quality which we adopted in this contribution consists of mak-

	Normal similarity	Adjusted similarity	*k*-means best precision	*k*-means best coverage	Proposed approach	LDA
Precision	0.6106	0.6130	0.6379	0.6321	0.6524	0.3368

Table 4.1: Precision results of TS-I*k*-NN on different baseline algorithms.

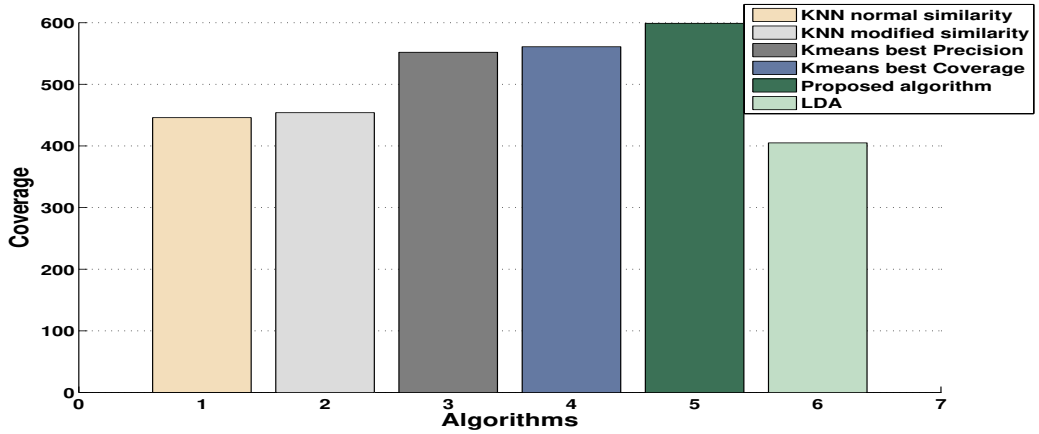


Figure 4.11: Coverage results on movielens 100K using a TS-I*k*-NN algorithm.

ing a balance between the users and the system benefits. Thus, we used both Precision and Coverage as accuracy and diversity evaluators.

Table 4.1 and Figure 4.11 show the results obtained using UCF algorithm with different configurations on a recommendation size of 10. The best results were proportional to the highest neighborhood size of the studied range and was equal to 50.

Precision results: the adjusted similarity measure has a negligible improvement in comparison to the conventional similarity measure. However, the optimization of the neighborhood has given a better result than the rest of the algorithms.

Coverage results, shown in Figure 4.11, are equally close when using both similarity measures and are better than LDA algorithm. However, the proposed approach gave the highest recommendation Coverage and is better than *k*-means algorithm.

Summary: In general, the results prove the potential of decreasing the search space extent using *k*-NN algorithm and also optimizing the neighborhood quality which enhances the recommendation quality.

4.4.4 Limitations and issues

In the previous section, we presented an evolutionary algorithm that acts in two stages with the aim of making a balance between Coverage and Precision. However, this

method suffers some limitations which we try to improve and solve:

- For a given user, the neighborhood is assigned according to binary weights (i.e., 0 or 1) which may exclude explorer users (i.e., the users who tend to rate rare and surprising items.). This issue can be solved by assigning to each user a real weight value (e.g., a weight in the range [0-1].).
- In the first stage, it is hard to fix the size of the neighborhoods, whereas big sizes allow a better exploration of the possible neighborhoods but increase the complexity.
- In the presented experiments, the correlation between the novelty of the recommendations and their Coverage is not cleared up, which involves additional work to show it up.

In the next section, we present a more sophisticated genetic-based solution to alleviate these issues.

4.5 An Evolutionary Based-Clustering Approach for Recommendation Diversification (GA-DCLUS)

Our proposed algorithm (genetic-based diversified clustering (GA-DCLUS)) consists of targeting a problem of two objectives: the scalability problem of MeCF algorithms, and the recommendation quality in terms of precision and diversity. The existing approaches address the scalability problem by partitioning the set of users into clusters of similar profiles using similarity measures, which may lead to very homogeneous clusters. In effect, two measures were used to validate the quality of clustering and are denoted by the intra-cluster and the inter-class distances. Intra-cluster aims to separate the clusters by maximizing the averaged distances between the centers of each cluster while inter-class maximizes the density of the clusters by minimizing the averaged distances between each pair of members inside the same cluster.

We have employed the evolution phenomenon of genetic algorithms with the intention of getting an overlapped clustering of the users, to address the mentioned problem.

We denote by U the set of users, and C the set of clusters. Our clustering scheme consists of creating a matrix $W = \{w_1, w_2, ..w_{|U|}\}$ of $|U|$ rows and $|C|$ columns. We define for each user $u \in U$, a belonging weigh vector $w_u \in W$ of $|C|$ values, which we denote by $w_u = \{w_{(u,1)}, w_{(u,2)}, ..w_{(u,|C|)}\}$, as to assign for u and a given cluster $j \in C$ a belonging weight value $w_{(u,j)}$. Furthermore, we assign each user u to one main cluster, and zero or more seconding clusters in respect to the belonging weight vector w_u. Next elements explain how we proceeded to assign users.

Main clusters set, which we denote by C^M, and it consists of $|U|$ value, as to have $C^M = \{c_1^M, c_2^M, ..c_{|U|}^M\}$, whereas each user u has a main cluster c_u^M in which we apply UCF algorithm to generate his recommendations. The cluster c_u^M is selected by identifying from w_u, the cluster of the highest belonging weight.

Secondary clusters set, which we denote by C^S, and it consists of $|U|$ sub-vector, as to have $C^S = \{c_1^S, c_2^S, ..c_{|U|}^S\}$. Each user u has a set c_u^S of n_u secondary clusters denoted by $c_u^S = \{c_1, c_2, ..c_{n_u}\}$. User u can participate in the clusters of c_u^S as a candidate neighbor only and does not get any recommendations. In order to improve the scalability, a minimum belonging threshold θ is added to the chromosome' encoding (i.e., the chromosome is the scheme of the clustering problem.). This setting is justified by two reasons: a) assigning the users to all or most clusters lead to very heavy clusters (i.e., clusters of big size.), which affects the scalability. b) the users of weak weights may not be selected during the neighborhood selection.

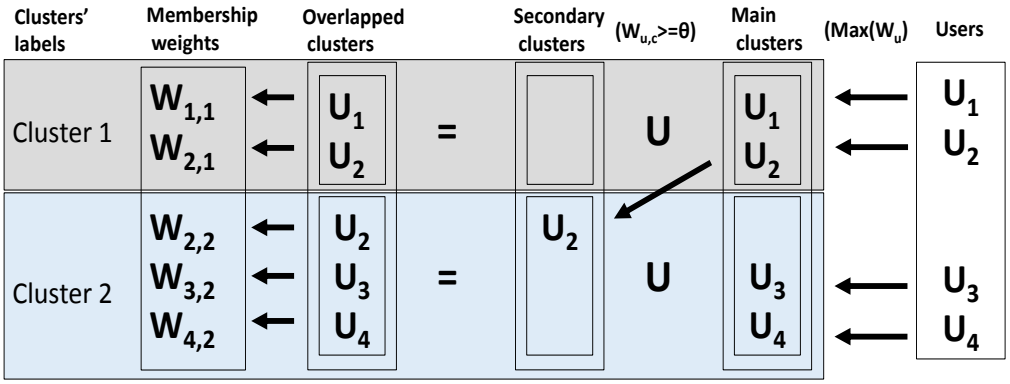


Figure 4.12: An illustrative example of our proposed overlapped clustering method

Figure 4.12 shows a simple illustrative example of the proposed overlapped clustering method, where $U = \{u_1, u_2, u_3, u_4\}$, and $|C|$=2. Our clustering algorithm consists of three steps:

- **Step 1**: We assign each user to one main cluster by identifying the index of the cluster with the highest membership score. In the current example, we see that the users u_1 and u_2 are assigned to the main cluster 1, whereas users u_3 and u_4 are assigned to the main cluster 2.
- **Step 2**: We check the membership weights again and compare them to the minimum membership weight θ. This step allows identifying the set of secondary clusters to which a particular user can be assigned. In the current example, we observe that the user u_2 is assigned to the secondary cluster 2.

- **Step 3**: The overlapped clusters are formed by the union of their corresponding main and secondary clusters, where each user has a different membership weight.

In UCF algorithm, two more questions are raised and are related to the number of clusters ($CLUS$) and the size of the neighborhood (NEI), which should be taken. On one hand, the clusters' number depends on the data structure, in which the ideal number is mostly fixed empirically [Berbague 18c] [Wang 14c]. On the other hand, the neighborhood size has an effect on the quality of the recommendations and depends directly on the computation complexity. In fact, the accuracy goes positively with the size of the neighborhood. In the works [Karabadji 18] [Berbague 18c], the number of neighbors was optimized by including this parameter in a genetic algorithm' encoding. We followed, in our proposition, a similar scheme that allows optimizing the number of clusters and the size of the neighborhood.

4.5.1 Encoding and decoding of chromosomes of GA-DCLUS

4.5.1.1 Encoding step of GA-DCLUS

Figure 4.13 shows the scheme of our chromosome which consists of two parts. The first part concerns the general clustering parameters and it holds three genes: the size of the neighborhood (NEI), the number of clusters (CLUS), and the minimum belonging threshold (TH). The second part represents the belonging vector W, which consists of $|U| \cdot |C|$ real value, one gene for each belonging weight.

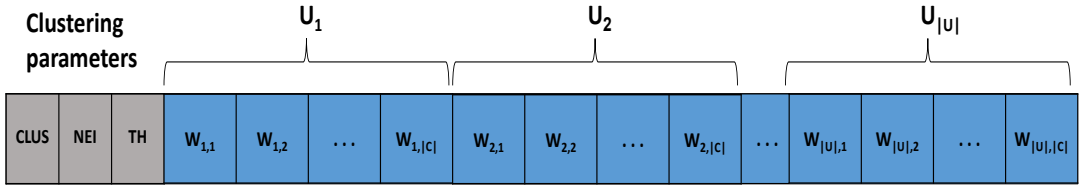


Figure 4.13: The chromosome encoding scheme of GA-DCLUS algorithm.

4.5.1.2 Decoding step of GA-DCLUS

The decoding step involves, firstly, getting the maximum number of clusters, the size of the neighborhood, and the minimum belonging threshold. Each user is, then, assigned to one or more clusters in respect to the minimum belonging threshold (i.e., for every user u that belongs to a cluster c, $w_{u,c} >= \theta$.). Then, among the considered clusters, the main cluster is selected regardless of the highest equivalent belonging weight while the rest of the clusters are considered secondary. The clusters' number, as well as the neighborhood size, are computed by considering the parameters max_C, min_C, max_N and

min_N which are the maximum clusters size, the minimum clusters size, the maximum neighborhood size, and the minimum neighborhood size respectively.

4.5.1.3 Example

We illustrate a decoding example of a small set of users with their belonging weights. We suppose that $U = \{u_1, u_2, u_3, u_4\}$ with a maximum and minimum clusters' size equal 2 (i.e., $min_C = 2.$) and 3 (i.e., $max_C = 3.$) respectively.

			U1			U2			U3			U4		
CLUS	NEI	TH	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$
2	48	0.8	0.95	0.85	0.7	0.55	0.92	0.88	0.90	0.81	0.91	0.7	0.94	0.74

Figure 4.14: Encoding example of GA-DCLUS algorithm.

Figure 4.14 illustrates a solution of a clusters' number equals 2, a neighborhood size equals 48, and a minimum belonging threshold equals 0.8. The illustrated configuration leads to ignore the third cluster, thus, the users may only belong to the cluster one (c_1) or the cluster two (c_2). In this way, we get: $c_1^S = \{c_1, c_2\}$, $c_2^S = \{c_2\}$, $c_3^S = \{c_1, c_2\}$, and $c_4^S = \{c_2\}$. As mentioned before, the users get their recommendations from their equivalent main cluster. This latter is selected in respect to the highest cluster weight. In this example, we have $c_1^M = \{c_1\}$, $c_2^M = \{c_2\}$, $c_3^M = \{c_1\}$ and $c_4^M = \{c_2\}$.

4.5.2 Selection of the best solution in GA-DCLUS

GA improves progressively the quality of the encoded solutions using evolution principles over the generations, whereas a fitness function is used to measure the progress of the optimization. GA involves the creation of an initial set of chromosomes denoted by *population* (POP). The initial population evolves for a predefined number of generations or till satisfying a stopping criterion. In each generation, a selection operator is employed to select a set of chromosomes denoted by *parents*, which will participate in generating a set of new individuals denoted by $offspring\ population$. This set is produced by performing genetic operators on each pair of parents to produce two children's chromosomes. Finally, the same selection operator is used to select among the initial population set and the offspring population, the population of the next generation.

4.5.2.1 Fitness function

The conventional clustering techniques aim to group together users with similar profiles. The similarity measures used in such clustering has two weakness. From one side, the recommendation coverage will be limited to a small portion of items, seen by the members of the same cluster. On the other side, the existing similarity measures are inefficient and subject to the data sparsity problem. We argue that good clustering should not only maximize the similarity inside the clusters but also helping users to share their rare and unique experiences by maximizing the diversity of users. In general, the efficiency of a recommender system may be evaluated by its ability at offering relevant and diverse recommendations. For this reason, we adopt both Precision and Coverage as two main quality metrics. Precision and Coverage are defined previously.

The rarity of an item represents a statistical indicator of its popularity. UCF tends to recommend popular items, while items of low consumption are less chance recommended [Park 08]. We define the rarity of an item as in the Equation 4.7:

$$rarity(i) = \frac{|U_i| - freq_{min}}{freq_{max} - freq_{min}}, \tag{4.7}$$

where, $freq_{max}$ and $freq_{min}$ denote longest and shortest items' profile lenght, while $|U_i|$ is the profile length of item i.

The tendency of users toward unpopular items may be personalized to define a pairwise measure between every two users. The objective of this metric is to amount the effect of a possible neighbor on a target user and to measure the number of rare items which may be recommended. Similarly to [Chatzicharalampous 15] in which explorer users were identified, we introduce a pairwise diversity to amount items' rarity which a user can offer to another one. This measure is computed as in Equation 4.8:

$$div(u_1, u_2) = \frac{1}{|I_2 - I_1|} \sum_{i \in I_2 - I_1} rarity(i), \tag{4.8}$$

where, u_1 and u_2 are two users defined by their profiles I_1 and I_2 respectively. Moreover, we define the cluster content as in Equation 4.9 to control the quality of the clusters in terms of the intra-class distance and diversity which are two conflicting measures.

$$cluster\_content(c) = \sum_{(u_1,u_2) \in c} \alpha \cdot sim(u_1, u_2) + (1 - \alpha) \cdot div(u_1, u_2), \tag{4.9}$$

where, c is a set of users that compose a cluster, and α is a controlling parameter in the range 0 to 1. Finally, we introduce the fitness function defined in Equation 4.10.

$$\begin{aligned} fitness\_function(ch) = \beta \cdot (1 - coverage) + (1 - \beta) \\ \cdot (\gamma \cdot (1 - precision) + (1 - \gamma) \cdot \frac{1}{|C|} \sum_{c \in C} cluster\_content(c) \end{aligned} \tag{4.10}$$

where, $|C|$ is the number of clusters, and β, γ are controlling parameters defined in the range 0 to 1.

4.5.3 Rating prediction and recommendation

In UCF, for every user, a set of neighbors are selected according to a similarity measure. These neighbors are used for rating prediction as in Equation 4.11.

$$Pr(u,i) = \bar{r_u} + \frac{\sum_{v \in_N (u)} sim(u,v) \cdot R(v,i)}{\sum_{v \in N(u)} sim(u,v)}, \tag{4.11}$$

where, $N(u)$ is the neighborhood set of user u, $\bar{r_u}$ is u' rating average and $sim(u,v)$ is the similarity between u and v.

The effect of the available similarity measures was discussed from different aspects such as the effect of the data sparsity and the effect of the prediction formula [Singh 18]. In effect, genetic algorithms and ant colony optimization were used to optimize the similarity between the users [Alhijawi 16a] [Parvin 19]. We acted similarly and used the equivalent belonging weights to adjust the similarity between users. The proposed clustering algorithm assigns to each user a set of clusters according to belonging weights. The similarity value between u_1 and a candidate neighbor u_2 is calculated as in Equation 4.12.

$$new\_sim(u_1,u_2) = \frac{w_{1,c_1^M} + w_{2,c_1^M}}{2} \cdot sim(u_1,u_2) \tag{4.12}$$

where, u_1 is the user whom we like to recommend items to, and $sim(u_1,u_2)$ is a conventional similarity measure between u_1 and u_2.

4.5.4 Improved prediction formula

We argue that the proposed clustering allows a better distribution of the items over the different clusters which may increase the items' coverage. However, the recommended items may keep being restricted to the popular ones only. We propose the use of a regularization weight, for each user, to tackle the dominance of popularity. The use of such weight allows indicating the level of popularity which will be assigned to users. The effect of such formulation leads to distribute more equally the recommended items over the whole users' set. We adjust Equation 4.11 by adding a regularization term as in Equation 4.13:

$$Pr_{imp}(u,i) = reg \cdot w_u^N \cdot Nov(i) + Pr(u,i), \tag{4.13}$$

where, w_u^N is a popularity controlling weight and reg is a regularization term which allows controlling the range of the adjustment effect. This parameter can take values

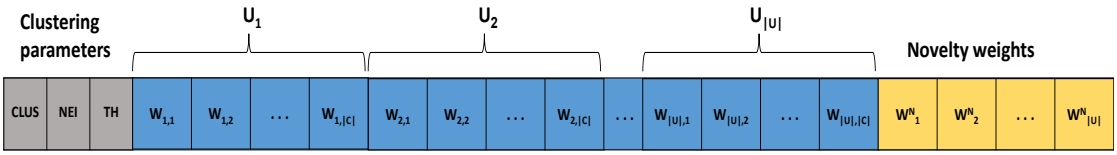


Figure 4.15: The chromosome encoding scheme of GA-DCLUS using novelty weights.

from 0 to 5. Finally, $Nov(i)$ is the normalized novelty of item i. The novelty [Castells 11] is related to the consumption frequency of an item over the set of users and it can be measured as demonstrated in Equation 4.14:

$$Nov(i) = -Log(\frac{|U_i|}{|U|}) \cdot Log(|U|), \tag{4.14}$$

We denote by W^N the novelty weights vector, which consists of $|U|$ value as to get $W^N = \{w_1^N, w_2^N, ...w_{|U|}^N\}$. The empirical optimization of W^N is an NP-hard problem search task. In effect, giving a low novelty weight may decrease the recommendations coverage, while increasing this weight value may lead to better coverage, but a worse precision. We propose to incorporate W^N as the third part in our chromosome encoding as illustrated in Figure 4.15. In the next algorithm, a summary of the optimization steps of our clustering algorithm.

Algorithm 1: Steps of running GA-DCLUS algorithm.

Data : train_set, users_set, population of chromosomes POP;
Result : users' main clusters C^M, users' secondary clusters C^S, users' popularity weights W^N, users' belonging weights W ;
while *(number of evaluations not reached)* **do**
  for $sol \in POP$ **do**
    (clus, nei, θ) $\leftarrow$ $get\_main\_paramaters(sol)$;
    for $u \in U$ **do**
      W $\leftarrow$ $W \cup get\_belonging\_weigths(u, sol)$;
      $W^N \leftarrow W^N \cup get\_popularity\_weight(u, sol)$;
      $C^M \leftarrow C^M \cup get\_main\_cluster(u, sol)$;
      $C^S \leftarrow C^S \cup$ get_secondary_clusters(u,sol,θ);
    $L_{recs} \leftarrow generate\_recommendations(train\_set, W, C^M, C^S)$;
    fitness $\leftarrow fitness \cup fitness\_function(train\_set, C^M, C^S, L_{recs})$;
  POP $\leftarrow optimize\_solutions(fitness, POP)$

4.5.5 Experimental work

4.5.5.1 Dataset

We have applied our experiments on movielens 100k. Additionally, we used 1M movielens dataset to examine the scalability of our algorithm. Both datasets were divided into 5 folds in a stratified manner. The training set was composed by collecting together 4 folds while the last fold was used for testing.

Dataset	#Users	#Items	#Ratings	Sparsity
Movielens 100k	943	1682	100000	6%
Movielens 1M	6040	3952	1000000	4%

Table 4.2: Dataset statics of MovieLens 100k and 1M.

We have analyzed the distribution of ratings over the set of items and users on 100k movielens dataset. The analysis shows a wide difference in the items' rating frequency. In effect, the items' rating average allows identifying a very small number of high-rated items against a larger number of few rated items. Figure 4.16 illustrates the distribution of ratings over the set of items.

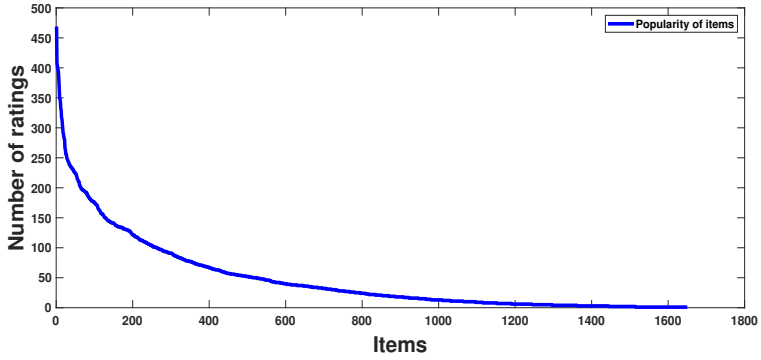


Figure 4.16: Popularity analysis of items in movielens 100K dataset.

Furthermore, the users' tendencies toward popular items can be calculated by summing the rarity of the items rated during the interaction with the system. Equation 4.15 introduces user's tendency:

$$tendency(u) = \frac{1}{|I_u|} \sum_{i \in I_u} 1 - rarity(i), \tag{4.15}$$

where, I_u is user u' profile. We have categorized the users by dividing the items' popularity into 20 levels, whereas each level is defined by a minimum and a maximum

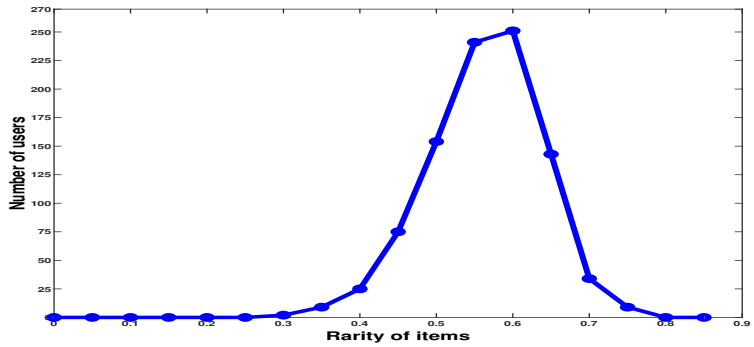


Figure 4.17: Illustration of users' tendencies toward popularity on movielens 100K.

threshold, then we counted for each level the number of users who have a popularity tendency inside this level, as illustrated in Figure 4.17. We observe that around 50% of users share similar tendencies toward popular items in the range 0.55 to 0.65. After this value, the number of users drops down.

To better demonstrate the correlation between users' pairwise similarity and rarity, we visualize the difference between these values for every user u. In more detail, we compute for each neighborhood set $N(u)$, the corresponding average similarity and average rarity (i.e., using Equation 5.) in two different cases: (a) maximizing the neighborhoods' intra-similarity (i.e., $Max - similarity$), and (b) maximizing the neighborhoods' intra-rarity (i.e., $Max - rarity$).

Figure 4.18 shows that maximizing the neighborhoods' similarity gives slightly higher rarity values over the set of neighborhoods (i.e., $Rarity|Max - similarity$), which may lead to low recommendation coverage. However, maximizing the neighborhoods' rarity has decreased significantly the neighborhoods' similarity (i.e., $Similarity|Max - Rarity$), thus generating irrelevant recommendations. In our proposal, we tune a combination parameter α to balance similarity and rarity in order to keep simultaneously good accuracy and coverage.

4.5.5.2 Baseline algorithms

Here, we compare our approach against UCF algorithms. We selected state-of-the-art algorithms that improve the quality of the neighborhood selection, such as clustering-based approaches.
We compare our approach to:

(i) k-NN algorithm ([Sun 05]) and four more variants of k-NN by adjusting Equation 4.11 as follows:

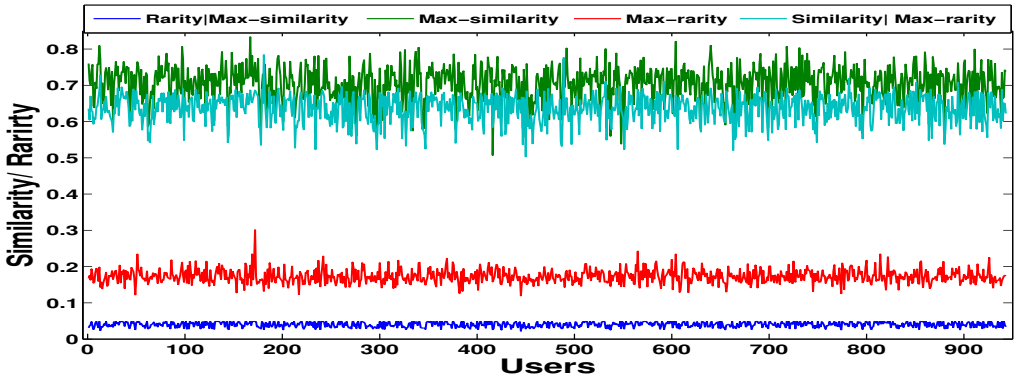


Figure 4.18: User-user pairwise similarty and rarity on 100K movielens dataset

(a) Random-based recommendation (k-NN$_{RAN}$) recommends randomly by picking items from the neighborhood members.

$$Pr_{RAN}(u,i) = random(max_{rate}), \tag{4.16}$$

where $random(max_{rate})$ is a function that generates a random value in the range $[1, max_{rate}]$.

(b) Average rating-based recommendation (k-NN$_{AVG}$) recommends by scoring items according to the average ratings given by the neighborhood members.

$$Pr_{AVG}(u,i) = \frac{\sum_{v \in N(u)} R_{v,i}}{|N(u)|} \tag{4.17}$$

(c) Popularity-based recommendation (k-NN$_{POP}$) recommends by scoring items according to their popularity in the neighborhood.

$$Pr_{POP}(u,i) = \frac{\sum_{v \in N(u)} \{1 | R_{v,i} \in R\}}{|N(u)|}, \tag{4.18}$$

(d) Inverse popularity-based recommendation (k-NN$_{IPOP}$) recommends by scoring items according to their inverse popularity in the neighborhood (i.e., items less rated by the neighborhood members).

$$Pr_{IPOP}(u,i) = 1 - Pr_{POP}(u,i), \tag{4.19}$$

Additionally, we compare our approach to the following clustering competitors:

(ii) k-means based approach] ([Zahra 15]) is a simple and fast clustering algorithm that consists of restricting the neighborhood selection in UCF algorithm on the level of the clusters instead of the whole users set. This algorithm consists of

maximizing iteratively the density by decreasing the distances between the centers and the members of each respective cluster. In each iteration, new centers are calculated then used for assigning the users accordingly. The main limitations of *k*-means are related to its stochastic initialization as well as the negative effect of the data sparsity on the similarity calculation.

(iii) Fuzzy *c*-means ([Birtolo 13]) is a particular clustering approach that allows a given user to belong to more than one cluster according to a membership degree.

(iv) Agglomerative hierarchical clustering (AHC) ([de Aguiar Neto 20]) is a bottom-up hierarchical clustering algorithm. AHC starts from each individual user as one separate cluster. Then, it progressively merges clusters together based on a linkage criterion. AHC stops running when the required number of clusters is reached or when no more clusters are possible.

(v) Gaussian mixture clustering (GMC) ([Yan 19]) is a probabilistic clustering method, which consists of a set of Gaussian distributions (i.e., one Gaussian for each cluster) where each one is defined by: (a) a mean which represents the center of the cluster, (b) a covariance matrix which defines the width of the cluster, and (c) a mixture probability which defines the size of the cluster. These parameters are optimized using maximum likelihood approach.

(vi) Self-organizing map clustering (SOM) ([Zhang 16]) is unsupervised neural network approach used for data reduction and visualization. SOM consists of a lattice of connected rectangular or hexagonal neurones where the output neurons layer has a size much lower than the original data. In training phase, a user is selected randomly and a winning neuron is identified based on its distance to the current user. The winning neuron and its neighborhood are then updated accordingly. New users' representations are obtained by passing original users vectors through SOM.

(vii) PCA-GAKM ([Wang 14b]) is a two steps algorithm proposed to deal with the initial seeds selection and the data sparsity problem. The algorithm reduces the rating matrix by using principal component analysis (PCA), then uses a genetic algorithm to decrease the intra-class distance, and then ends-up by resorting to *k*-means.

4.5.5.3 Experimental parameters of GA-DCLUS

In the aim of avoiding time consumption, we encoded our chromosomes in the form of real value variables instead of the binary representation which is a transformation task of high time-consuming. We have initialized our genetic algorithm by creating randomly 50 chromosomes, to ensure the diversity of the individuals and to get a

good distribution over the search space. Additionally, we used a polynomial mutation [Deb 96] which simulates bit-flip mutation and we set the mutation probability to be 0.9. As a crossover operator, we used the simulated binary crossover (SBX) [Deb 95] with a probability equals 0.02. During the evolution process which lasts 250 iterations, a binary tournament selection strategy is used to select new individuals.

4.5.5.4 The experimental scenario and configuration

The results were gotten by changing the number of clusters from a small number starting from min_C to max_C. Also, we followed the same scenario to previous works [Wang 14c][Karabadji 18] and chose to analyze the neighborhood size effect on a small range on the extent from min_N to max_N. The results were validated by incrementally increasing the neighborhood size by 10 every time. Finally, we averaged the results of 10 times executions of the algorithms on the different neighborhood configurations, then we measured the recommendations performances for the top-10 recommendations and selected the best results of each algorithm.

4.5.5.5 Evaluation metrics

In this section, we validate the results of each algorithm in terms of the precision and the diversity of the recommendations. We used Precision defined previously in addition to Recall and F1 measures. Furthermore, we adopted Coverage defined and Novelty metrics defined previously.

4.5.5.6 Results and discussion

Our aim, in this section, is to validate the performance of our algorithm using different configurations. We have set the parameters of our algorithm as shown in Table 4.3.

reg	max_N	min_N	max_C	min_C	α	β	γ
[0-5]	50	10	50	4	0.2	0.9	0.1

Table 4.3: Experimental parameters of GA-DCLUS algorithm.

4.5.5.7 Analyzing the similarity weight effect

We present the performance of our algorithm in terms of the employed similarity measure. For this end, we have run our algorithm 10 times with two configurations: **a)** GA using a normal similarity measure (GA_{sim}). **b)** GA using a weighted similarity measure

(GA_{wsim}). In every run, we have stored the value of the fitness function along with 40 generations.

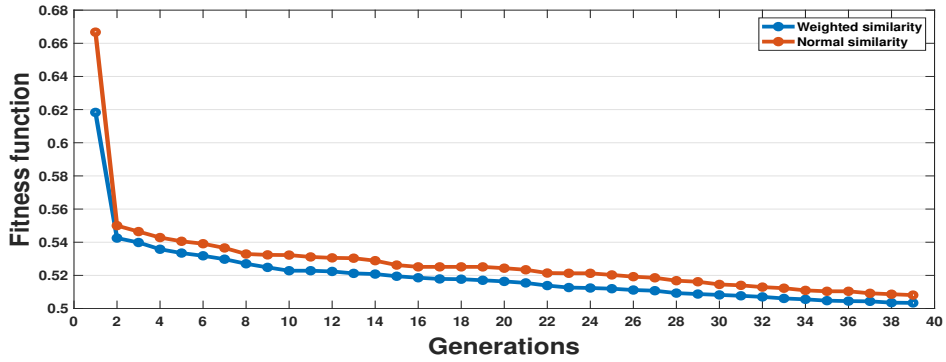


Figure 4.19: Convergence speed of GA-DCLUS with normal and weighted similarity measure.

Figure 4.19 demonstrates the convergence of our algorithm in both cases, whereas GA_{wsim} starts with a significantly better fitness value after just one generation due to the initial neighbor selection which is independent of the statistical similarity measure affected by the data sparsity. GA_{wsim} keeps converging faster than GA_{sim}, when GA_{wsim} reaches the value 0.52 in the generation 14, GA_{sim} reaches the same value in the generation 22 (i.e., after 9 generations.) which is 450 additional evaluations.

Furthermore, we tested the effect of the employed similarity on the recommendation quality in terms of relevancy metrics (i.e., Precision, Recall, and F1.) and diversity metrics (i.e., Coverage, and Novelty.). Results of both GA_{sim} and GA_{wsim} are listed in Table 4.4.

Configuration	Precision	Recall	F1	Coverage	Novelty
GA_{sim}	0.5560	**0.6831**	0.6130	0.4713	0.8165
GA_{wsim}	**0.5625**	0.6830	**0.6269**	0.4733	0.8165

Table 4.4: Comparison of GA-DCLUS results using different similarity measures.

The obtained results show similar performances when using both similarities with a slight superiority of GA_{wsim} against GA_{sim}, hence we adopted GA_{wsim} in the rest of the experimental work due to its better performance.

4.5.5.8 Algorithm evaluation

GA converged toward high neighborhood size in the range between 45 and 50. This proves a positive correlation between the size of the neighborhood and the quality of

recommendations. Furthermore, the number of clusters has tended to the maximum value in the range between 35 and 50. This confirms that GA is able to generate better clusters quality when we increase max_C. We validated the quality of results when changing *reg*' value. Precision, Recall, F1, Coverage, and Novelty. Results are listed in Table 4.5.

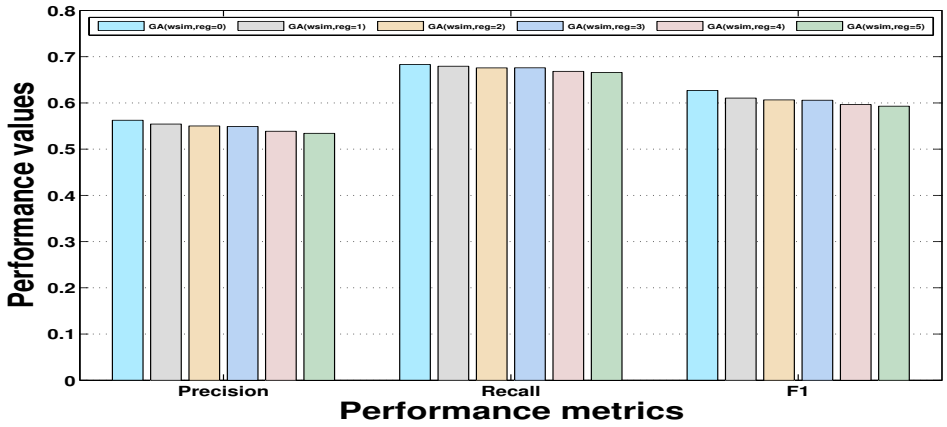


Figure 4.20: GA-DCLUS performance with different configurations versus the relevancy metrics on 100k movielens dataset.

Configuration	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=0}$	**0.5625**	**0.6830**	**0.6269**	0.4733	0.8165
$GA_{wsim,reg=1}$	0.5543	0.6795	0.6105	0.4878	0.8205
$GA_{wsim,reg=2}$	0.5502	0.6759	0.6066	0.5048	0.8249
$GA_{wsim,reg=3}$	0.5490	0.6760	0.6059	0.5214	0.8310
$GA_{wsim,reg=4}$	0.5387	0.6683	0.5965	0.5454	0.8347
$GA_{wsim,reg=5}$	0.5343	0.6660	0.5929	**0.5533**	**0.8384**

Table 4.5: Comparison of GA-DCLUS with different configurations on 100k movielens dataset.

Table 4.7 shows that incrementally increasing *reg* value from 0 to 5 has given better results in terms of novelty and coverage of recommendations with a small loss of precision from 0.56 to 0.53 when $reg = 5$. These results are further illustrated in Figure 4.20 and Figure 4.21, whereas we observe a gradual descent in relevancy metrics and a gradual rise in diversity metrics. This experience proves the ability to control the recommendation coverage versus the precision of the recommendations by changing *reg* value accordingly.

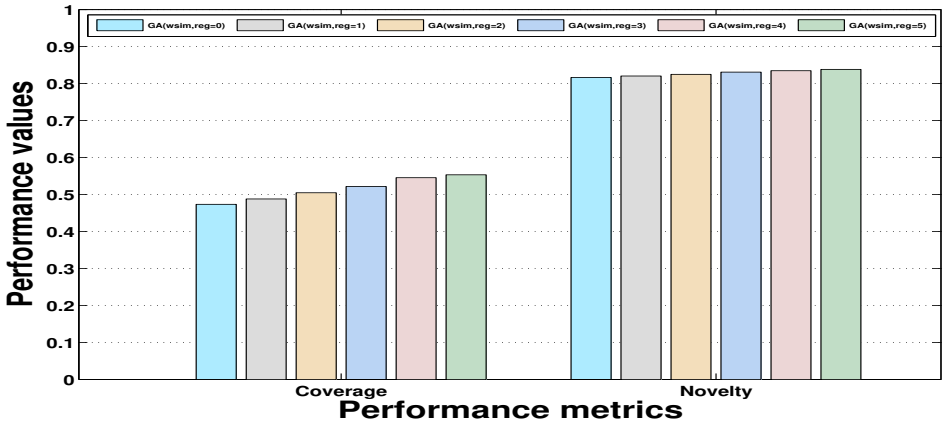


Figure 4.21: GA-DCLUS performance with different configurations versus the diversity metrics on 100k movielens dataset.

Algorithm	Precision	Recall	F1	Coverage	Novelty
GA_{wsim}	**0.563**	**0.683**	**0.627**	0.473	0.817
k-NN	0.437	0.512	0.472	0.300	0.766
k-NN$_{RAN}$	0.455	0.579	0.509	**0.500**	0.809
k-NN$_{AVG}$	0.500	0.604	0.547	0.419	0.798
k-NN$_{POP}$	0.480	0.592	0.511	0.357	0.782
k-NN$_{IPOP}$	0.451	0.574	0.505	0.493	**0,822**

Table 4.6: Comparison of GA-DCLUS to k-NN variants on 100k movielens dataset.

Table 4.6 lists the results in terms of Precision, Recall, F1, Coverage, and Novelty of our GA in comparison to different k-NN variants. By comparing results of k-NN$_{POP}$ and k-NN, we notice again the correlation between the popularity and accuracy. However, ranking recommendation according to their Novelty in k-NN$_{IPOP}$ (i.e., inverse popularity) has a small Novelty difference in comparison to GA_{wsim}. Generally, the different variants gave better performances in comparison to k-NN, which confirms the ability of our similarity in creating better quality clusters.

4.5.5.9 Comparision to other methods

Table 4.7 lists results in terms of Precision, Recall, F1, Coverage, and Novelty of our GA when $reg = 0$ and $reg = 5$ in comparison to baseline recommenders described previously. We mention that $GA(wsim, reg = 0)$ gives the best Precision and the less Coverage, while $GA(wsim, reg = 5)$ gives the best Coverage and the less Precision.

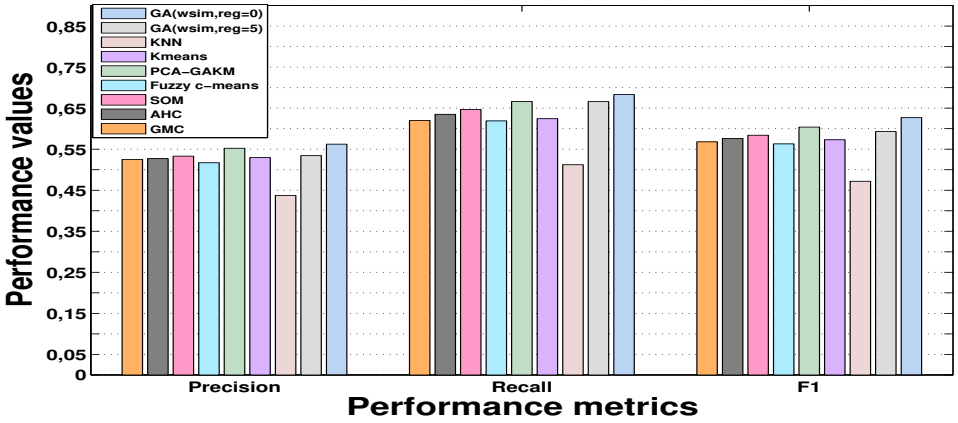


Figure 4.22: GA-DCLUS comparison to other methods on base of relevancy metrics on 100k movielens dataset.

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=0}$	**0.563**	**0.683**	**0.627**	0.473	0.817
$GA_{wsim,reg=5}$	0.534	0.666	0.593	**0.553**	**0.838**
k-means	0.530	0.624	0.573	0.413	0.799
$Fuzzy$ c$-means$	0.517	0.619	0.563	0.399	0.795
SOM	0.533	0.647	0.584	0.444	0.809
AHC	0.527	0.635	0.576	0.426	0.803
GMC	0.525	0.620	0.568	0.4241	0.796
PCA-$GAKM$	0.552	0.666	0.604	0.452	0.811

Table 4.7: Comparison of GA-DCLUS to other methods on 100k movielens dataset.

A first observation can be noticed when comparing k-NN results to k-means, whereas sharing neighbors in k-NN is associated with low items coverage (i.e., high neighborhood intersection between users due to the openness of neighbors selection.). We also observed by comparing k-means results to $PCA - GAKM$ that better precision is associated with higher coverage, which confirms that users differ in their tastes toward rare items and express positive preferences.

Results illustrated in Figure 4.22 and Figure 4.23 show that our GA is able to manage the assignment of users to different overlapped clusters and conserve better precision and coverage than the rest of the algorithms. Even that GA generates overlapped clusters, its coverage is better than k-NN due to the fitness function which collects a cluster's members not only based on the similarity but also the pairwise diversity. $PCA - GAKM$ is the best competitor clustering algorithm however its recommendation coverage is lower than $GA_{wsim,reg=0}$.

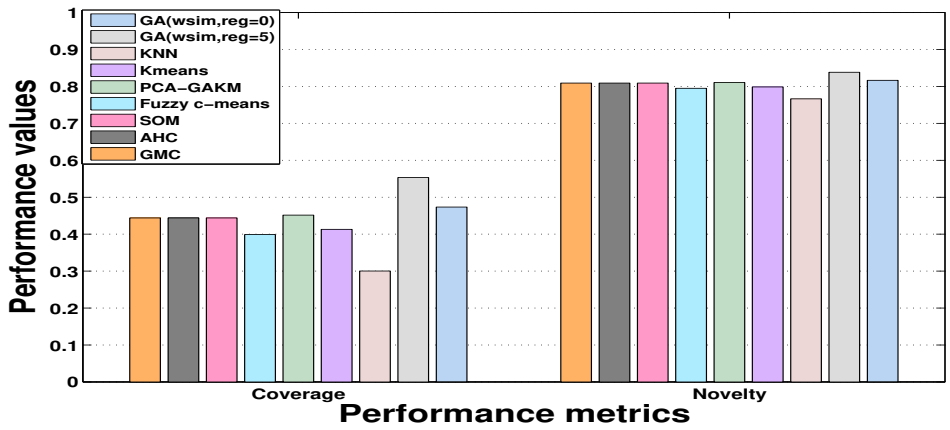


Figure 4.23: GA-DCLUS comparision to other methods on base of diversity metrics on 100k movielens dataset.

4.5.5.10 Results on the 1M movielens dataset

In this section, we evaluate the performance of our algorithm when we scale up our algorithm to the 1M movielens dataset. Table 4.8 lists GA results in terms of Precision, Recall, F1, Coverage, and Novelty against baseline competitors.

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=0}$	**0.636**	**0.589**	**0.611**	0.535	0.843
$GA_{wsim,reg=5}$	0.609	0.578	0.593	**0.631**	**0.872**
k-NN	0.458	0.383	0.417	0.295	0.781
k-NN$_{RAN}$	0.361	0.390	0.375	0.527	0.844
k-NN$_{AVG}$	0.514	0.475	0.493	0.460	0.822
k-NN$_{POP}$	0.520	0.476	0.497	0.429	0.802
k-NN$_{IPOP}$	0.471	0.455	0.296	0.524	0.848
k-means	0.576	0.468	0.517	0.461	0.798
$Fuzzy$ c$-means$	0.517	0.455	0.484	0.400	0.808
SOM	0.555	0.499	0.526	0.437	0.821
AHC	0.545	0.476	0.508	0.458	0.814
GMC	0.548	0.468	0.505	0.461	0.812
PCA-$GAKM$	0.604	0.518	0.558	0.479	0.820

Table 4.8: Comparison of GA-DCLUS to other methods on 1M movielens dataset.

Our experimental work on 1M movielens dataset has considered a bigger range of cluster's number that varies in the range [4- 100], whereas the results confirm the ability

to scale up our algorithm, whereas best results depend on high clusters' number near to 100 for $GA(wsim)$. In effect, our proposed GA kept giving better results in terms of relevancy metrics and diversity metrics, which confirms that GA is able to improve the quality of clusters even when the search space is more complex.

4.5.5.11 Comparision against recommendation re-ranking

MMR re-ranker: we use MMR re-ranker algorithm described in Section 2.2 for comparison. To this end, for each $u \in U$, we have generated a recommendation set R_u, which we improved using the MMR algorithm to generate a final recommendation set S_u. We have used the next objective function:

$$i^* = argmax_{i \in (R_u - S_u)}(1-\lambda) \times rel(i, S_u) + \lambda \times gnov(i, S_u), \tag{4.20}$$

where, λ is a controlling parameter fixed empirically to 0.2, and $rel(i, S_u)$ is relevancy of i in respect to S_u' items and is defined as:

$$rel(i, S_u) = \frac{Pr(u,i) + \sum_{j \in S_u} Pr(u,j)}{1 + |S_u|}, \tag{4.21}$$

Moreover, gnov(i,S_u) is the global novelty of S_u after adding i, and is defined as:

$$gnov(i, S_u) = \frac{Nov(i) + \sum_{j \in S_u} Nov(j)}{1 + |S_u|}, \tag{4.22}$$

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=3}$	**0.549**	**0.676**	**0.606**	**0.521**	**0.831**
k-mean$_{(MMR)}$	0.525	0.618	0.570	0.442	0.804
$Fuzzy$ c$-means_{(MMR)}$	0.513	0.617	0.560	0.423	0.803
$SOM_{(MMR)}$	0.527	0.644	0.580	0.472	0.817
$AHC_{(MMR)}$	0.522	0.632	0.572	0.458	0.811
$GMC_{(MMR)}$	0.521	0.618	0.565	0.456	0.805
$PCA\text{-}GAKM_{(MMR)}$	0.546	0.665	0.600	0.484	0.820

Table 4.9: Comparision of GA-DCLUS to MMR on 100k movielens dataset.

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=5}$	**0.609**	**0.578**	**0.593**	**0.631**	**0.872**
k-means$_{(MMR)}$	0.559	0.435	0.489	0.475	0.804
$Fuzzy\ \text{c}-means_{(MMR)}$	0,514	0.453	0.481	0.422	0.816
$SOM_{(MMR)}$	0.551	0.497	0.522	0.465	0.832
$AHC_{(MMR)}$	0.540	0.474	0.505	0.494	0.824
$GMC_{(MMR)}$	0.543	0.466	0.502	0.494	0.822
$PCA\text{-}GAKM_{(MMR)}$	0.599	0.537	0.566	0.473	0.835

Table 4.10: Comparison of GA-DCLUS to MMR on 1M movielens dataset.

Tables 4.9 and 4.10 lists Precision, Recall, F1, Coverage, and Novelty of our GA against MMR algorithm applied on k-means and $PCA - GAKM$ algorithms. These results demonstrate the superiority of our algorithm in comparison to MMR, due to the optimization of a different controlling weight w_u for each user u, rather than one parameter λ for all users in case of MMR. Furthermore, rating prediction, in the case of our algorithm, is optimized in parallel with the optimization of the clusters' quality, which reduces the precision drop. However MMR re-ranks the recommendations only after creating the clusters.

xQuAD re-ranker: we use xQuAD re-ranker algorithm described in Section 2.2 for comparison. To this end, we labeled the items with popular, and rare features using a popularity threshold equivalent to 5% of the users' number (i.e., items rated by more than 5% of the users were considered popular, while the rest of the items were considered rare).

Tables 4.11 and 4.12 report Precision, Recall, F1, Coverage, and Novelty for our GA against xQuAD algorithm applied on baseline clustering algorithms. xQuAD results have a slight difference to MMR in terms of relevance metrics with a slight loss in Precision, and Recall. Even xQuAD has higher Coverage in comparison to MMR on both datasets, our proposed GA still has higher performances.

In conclusion, the results on two different datasets show a positive correlation between Novelty and Coverage and a negative correlation with Precision. Moreover, $GA_{wsim,reg=0}$ gives better results in comparison to other algorithms. In Particular, diversity metrics can be improved by adjusting the prediction formula, which can be explained by the dominance of popular items when the recommendation is based on the use of similarities during the rating prediction.

4.5.5.12 Summary

The quality of the recommendations is subject to two conflicting objectives: the recommendation relevancy, and the recommendation diversity. Recommendation relevancy

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=3}$	**0.549**	**0.676**	**0.606**	**0.521**	**0.831**
k-mean$_{(xQuAD)}$	0.523	0.618	0.566	0.468	0.801
$Fuzzy$ c$-means_{(xQuAD)}$	0.512	0.617	0.560	0.442	0.799
$SOM_{(xQuAD)}$	0.526	0.643	0.579	0.499	0.813
$AHC_{(xQuAD)}$	0.520	0.632	0.570	0.485	0.807
$GMC_{(xQuAD)}$	0.517	0.616	0.562	0.481	0.801
$PCA\text{-}GAKM_{(xQuAD)}$	0.544	0.664	0.598	0.505	0.817

Table 4.11: Comparison of GA-DCLUS to xQuAD on 100k movielens dataset.

Algorithm	Precision	Recall	F1	Coverage	Novelty
$GA_{wsim,reg=5}$	**0.609**	**0.578**	**0.593**	**0.631**	**0.872**
k-means$_{(xQuAD)}$	0.554	0.434	0.487	0.508	0.803
$Fuzzy$ c$-means_{(xQuAD)}$	0.512	0.453	0.481	0.453	0.813
$SOM_{(xQuAD)}$	0.548	0.496	0.521	0.494	0.830
$AHC_{(xQuAD)}$	0.537	0.473	0.503	0.520	0.822
$GMC_{(xQuAD)}$	0.540	0.465	0.499	0.519	0.820
$PCA\text{-}GAKM_{(xQuAD)}$	0.596	0.537	0.565	0.499	0.833

Table 4.12: Comparison of GA-DCLUS to xQuAD on 1M movielens dataset.

is critical to validate the recommendations. However, recommendation diversity is a new important quality criterion that is negatively correlated to relevancy. MeCF algorithm is one of the simplest recommendation techniques which was under many enhancements. This technique suffers two weaknesses: the long tail problem and the scalability problem. Thus, we proposed a new genetic-based clustering formulation that makes a balance between the relevancy and the diversity of the recommendations while improving the scalability. The proposed clustering algorithm allows optimizing the number of clusters as well as their quality, whereas each user is represented by a belonging weights vector, and a popularity controlling weight. According to the belonging weights vector, a given user is assigned one main cluster from which he gets his recommendations and a set of secondary clusters where he only participates in the recommendation process. Furthermore, the popularity of controlling weight allows decreasing the long tail effect with a small loss in precision. Experimentations on movielens benchmarks showed the superiority of our proposed clustering to competitors in terms of relevancy and diversity metrics. In future works, we aim to study how users' preferences change over time; to deliver better-personalized recommendations. An interesting solution would be to look for a formal description of users preferences and their tolerance toward diversity and novelty.

4.6 Conclusion

MeCF algorithms are among the simplest recommendation algorithms in the domain. However, these algorithms suffer many limitations such as the sparsity, the scalability, the cold start problem, and the poor recommendation diversity. In this thesis, we investigated the use of evolutionary algorithms to alleviate these limitations. In the first step, we proposed a genetic algorithm scheme to handle the selection of the initial seeds in k-means algorithm. As the proposed scheme gave encouraging results, we continued the evaluation of the algorithm by analyzing the recommendation diversification possibility by adapting the similarity measures, then incorporating this measure within a clustering algorithm. As a result, combining an adapted similarity measure and a clustering algorithm proved its positive impact on the recommendation quality, the scalability rather than the recommendation re-ranking process.

Information is the resolution of uncertainty.

Claude Shannon

5

A personalized neighborhood selection to improve novelty.

▷ *In this chapter, we develop a MeCF algorithm, by linearly combining two probabilistic criteria for selecting the right neighborhood of a target user and provide him/her accurate, novel, and diversified item recommendations. The combination of these two probabilistic quality measures forms a fitness function, which guides the evolution of a genetic algorithm. For each target user, the genetic algorithm explores the users whole search space and selects the most suitable neighborhood for him. Thus, our approach makes a balance between the accuracy and the novelty of the provided item recommendations.* ◁

Chapter outline

5.1 Introduction

Recommender systems aim at satisfying clients' needs by proposing them relevant items that presumably match their preferences [Jannach 10]. Collaborative filtering is considered to be the most prominent recommendation paradigm that harnesses the opinions and traces of other users to accurately identify different taste categories and items, in which the user may be interested in. In the case of UCF [Bobadilla 13], the neighborhood for each user is identified based on his/her similarity with others. Based on this neighborhood of users, we can predict the target user's ratings over items. The proximity of neighbors to the target user highly affects the items' rating prediction. However, conventional recommender systems do not take the novelty that a neighbor may bring to the target user into consideration. That is, the similarity measures capture only the relevance of two users in terms of the co-rated items, and do not consider the contribution of a user in terms of novel and potentially serendipitous items that are not co-rated with the target user. We therefore want to provide both accurate and diversified recommendations by enhancing the neighbors' selection in UCF. We propose a new neighborhood selection approach, which is based on a genetic algorithm that uses a fitness function of two different criteria to capture for each pair of users, their relateness, their diversity, and the novelty which they may bring to each other. Our aim is to optimize, for each user, his/her suitable neighborhood of users which may improve the accuracy, the diversity and the novelty of recommendations. In UCF, the neighborhood selection has a crucial impact on the quality of the generated recommendations. That is, by improving the neighborhood selection, we can improve the accuracy, novelty and diversity of recommendations. The our contribution is about three-folds:

- We propose a neighbourhood of users for a target user by taking under consideration the two criteria relevance and novelty.
- We use a genetic algorithm to optimize a fitness function which linearly combines the two aforementioned criteria.
- We compare our method with other comparison partners and show that we can provide both accurate and novel recommendations.

In the same direction like [Zhang 09], we try to provide both accurate and novel recommendations. Our proposed method aims to identify for each target user, those candidate neighbors who will improve both the accuracy and the novelty of the item recommendations.

5.2 An evolutionary approach to improve neighborhood selection

The efficacy of UCF algorithms depends on the similarity measure which indicates for every target user his/her corresponding neighbors. Inspired by the work of

[Karabadji 18], we seek to explore the search space which consists of all possible sets of users, to select among them the best neighborhood. In particular, we propose a genetic algorithm, which is able to select for each target user a suitable neighborhood of similar users, for providing more accurate, novel and diversified item recommendations, by combining two criteria (relevance, and novelty). In our model, we try to optimize the neighborhood size, as well as, which candidate members will be selected to be part of the neighborhood. In particular, the neighborhood size k can be fixed with a minimum and a maximum threshold. Thus, the whole search space N can be computed as shown by Equation 5.1.

$$N = \sum_{k=min}^{max} C_{|U|}^{k} = \sum_{k=min}^{max} \frac{|U|!}{|U|!(|U|-k)!}, \tag{5.1}$$

where k is the size of the neighborhood, C the number of all possible combinations of k, and U the set of users. Evolutionary algorithms seem to be a good solution for such a big search space, since they have the ability to explore a wide range of possible solutions due to their random initialization and selection strategies.

Genetic algorithms randomly create an initial set of chromosomes. The representation of a chromosome may take many forms (e.g., binary values, real values, etc.) and it is represented in a sequence of genes, whereas each gene consists of a subset of variables, as shown in Figure **??**. The initial chromosomes evolve over the generations by genetic operations such as selection, mutation, and crossover, to optimize their quality. We aim at getting recommendations that satisfy conflicting quality measures. Hence, we use a fitness function that consists of a linear combination of two measures (i.e. relevancy, and novelty). In our proposed method, this process is repeated until our fitness function ceases to improve or a maximun number of steps is reached. This optimization process can be performed for each user independently. As far as the other two genetic operations are concerned, the crossover operation always produces two new chromosomes based on the two parent chromosomes by exchanging their genes, whereas the mutation operation randomly changes the values of a chromosome.

For GA optimization, we pick one target user, and randomly create an initial set of chromosomes of a binary representation. These chromosomes evolve n times, whereas in each generation the mutation and crossover operators are applied on the population to produce new chromosomes. The new population which forms the new generation is selected, regardless of the fitness function. This latter guides the optimization of the users' neighborhood and improves the user relevance, and item novelty which the members of this neighborhood bring to the target user.

5.2.1 Neighborhood encoding

Similarity measures such as cosine similarity, pearson correlation, etc. capture only few aspects of the notion of similarity over the possible candidates in the users' set and they do not ensure a global view over the quality of the whole neighborhood. For this

reason, inspired by the solution proposed in [Karabadji 18], we represent each chromosome with three genes, as it is shown in Figure 5.1. The first gene holds the size of the neighborhood set. The second gene denotes the first member in the neighborhood. Finally, the third gene represents the identifier of the whole neighborhood set. Some indicative examples of genes are shown in Figure 5.1 with possible binary representations of this encoding. Further details are available in [Karabadji 18]. The use of a GA during the neighborhood optimization allows to explore the search space independently from the data structure. It also allows to overcome the limitation of only observing the pairwise measurements between users (i.e., the target user and each of his/her neighbors) to assess the whole set of neighbours.

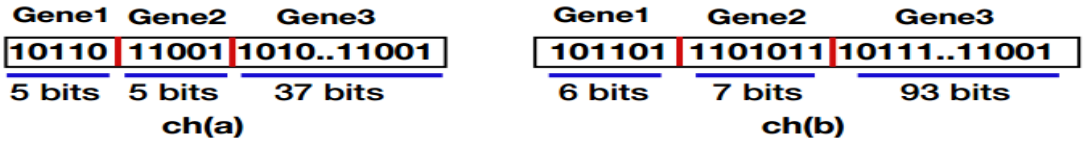


Figure 5.1: Chromosome encoding examples of the neighborhood optimization algorithm.

5.3 Criteria for Building the User Neighborhood

In this Section, we describe the two criteria we set for forming the user's neigboorhood. In recommender systems, the recommendation process is based on the users' ratings. This formally is expressed by using the equation $U \times I \rightarrow R$, where U is the set of users, I is the set of items and R the set of ratings given by users to the items. Please notice that in a recommender system for movies, each movie may also be described by a set of additional features such as genre, year of release and participants of the movie. Furthermore, we denote with u_t, u_c the target user and his candidate neighbor, respectively. To ease the discussion for the creation of the user-item profile, we will use a toy example, as shown in Table 5.1.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	-	-	-	-	-	-	5	1
u_2	1	4	-	-	-	-	4	2
u_3	-	-	4	2	-	-	5	4
u_4	-	-	3	4	4	5	5	4

Table 5.1: Example of the user-item rating matrix.

5.3.1 Relevance criterion

In collaborative filtering, an effective similarity measure allows to select a suitable set of neighbors which can predict accurately the users' preferences. For the sake of improving the recommendation quality, we propose a similarity measure based on the users' rating information. The basic idea is to select among the users' set, those users who share the same positive or negative preferences. In other words, for every co-rated item between the target user and his candidate neighbor, we compute the difference in the rating values and count only the items which have a difference value lower or equal to 1 (e.g., the rating 5 and 4, 4 and 3, etc.). In the same direction with [Bellogín 14], since the number of co-rated items can be small and consequently affect the similarity effectiveness, we set for any two users u_t and u_c a confidence probability threshold, as it is shown by Equation 5.2:

$$P_{conf}(u_t, u_c) = \frac{|\{i \in I | R_{c,i} \neq 0; R_{t,i} \neq 0\}|}{|I|}, \tag{5.2}$$

where $R_{c,i}$ is the rating given by the user u_c to the item i and I is the set of all items. Then, the similarity between any two users is computed as it is shown in Equation 5.3:

$$P_{rel}(u_t, u_c) = P_{conf}(u_t, u_c) \times \sum_{i \in (I_{u_c} \cap I_{u_t})} \frac{|\{i | R_{t,i} - R_{c,i}| <= 1\}|}{|I_{u_c} \cap I_{u_t}|}, \tag{5.3}$$

where $R_{c,i}, R_{t,i}$ are ratings given by u_t and u_c to item i, respectively. I_{u_c}, I_{u_t} are those items, which were rated by u_c and u_t, respectively.

In our running example, for the set of users $\{u_1, u_2, u_3, u_4\}$, we consider their preferences on the set of items $\{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8\}$, using a rating scale in the range [1-5] as shown in the Table 5.1. Based on Equation 5.3, the most relevant user to the target u_1 is u_2 because their $P_{rel}(u_1, u_2) = 0.25$, whereas the revelance of u_1 with u_3 is 0.12 and the revelance of u_1 with u_4 is 0.12.

5.3.2 Novelty criterion

For a recommender system that has to rely only on a user-item rating matrix (without any other information about categories of items, domains of users' interests, etc.), the simple popularity-based novelty definition is suitable. Thus, novelty can be defined as the opposite of popularity, which means that an item is more novel if fewer people are aware of it. In other words, novelty corresponds to the probability that an item has been rated or had any other type of interaction with a user.

$$novelty(i) = nov(i) = -\frac{|U_i|}{|U|} \tag{5.4}$$

where U_i is the set of users that rated item i, and U is the set of all users.

Based on Equation 5.4, an item can be considered as more novel, if the users have less interacted with it (i.e., it received less ratings, or it is less frequently purchased or viewed). In order to highlight the existence of highly novel items (favoring few very novel items and penalizing many less novel items), we can consider the logarithm of the item's novelty, as it is shown in 5.5.

$$nov(i) = -log_2 \frac{|U_i|}{|U|} \tag{5.5}$$

where U is the users set and U_i the set of users who have rated item i. Then, we define the novelty criterion as the sum of products of all possible items which the neighborhood can recommend to the target user as shown in Equation 5.6:

$$P_{nov}(u_t, u_c) = \frac{1}{|I_{u_c} - I_{u_t}|} \sum_{i \in (I_{u_c} - I_{u_t})} \frac{R_{u_{c,i}}}{M}(nov(i)), \tag{5.6}$$

where $R_{u_{c,i}}$ is the rating given by the user u_c to the item i and M is the maximum value in the rating range. Based on Equation 5.6, user u_4 is closest (in terms of novelty) to the target user u_1 because their $P_{nov}(u_1, u_4) = 0.64$, whereas the novelty between u_2 and u_1 is 0.525, and the novelty between u_3 and u_1 is 0.3.

5.4 Defining and Optimizing our Fitness Function

In this Section, we define the fitness function and describe the steps of the recommendation process. Our neighborhood selection method, incorporates 2 quality measures. Firstly, the relevance criterion is used to capture the affinity between the target user and his neighbors. Next, the novelty criterion can be used to identify those users that can bring novel and diversified recommendations to the target user. We compute the product of the two aforementioned criteria, after we have normalized their values, as it is shown in Equation 5.7.

$$P = P_{rel} \cdot P_{nov}, \tag{5.7}$$

where P is the final user-user similarity matrix. the genetic algorithm uses a simple and fast fitness function, which allows a quick search of the solutions space. Thus, we are able to compute for each pair of users their relevance, and the novelty which

they bring to each other and store this information in the memory. Then, during the evolution of the genetic algorithm we use this information for guiding the search.

Algorithm 5.1: Steps of neighborhood optimization.

Data : user-item rating matrix R, set of users U, set of items I, train_set, test_set;
Result : users' neighborhood sets
for $(u_1, u_2) \in U$ **do**
    pairwise_relevance $\leftarrow$ ComputeRelevanceCriterion(u_1, u_2) ;
    (using Eq.5.3)
    pairwise_novelty $\leftarrow$ ComputeDiversityCriterion(u_1, u_2) ;
    (using Eq. 5.6)
for $u \in U$ **do**
    while *n steps not reached* ***AND*** *fitness function continues to improve* **do**
        $nei_u \leftarrow GeneticAlgorithmsoptimization(pairwise\_accuracy, pairwise\_novelty, train\_set)$;
        (using Eq.5.7).
    $l_u \leftarrow recommend(u, nei_u, R)$;
    (using Eq.5.8)

5.5 Items' Rating prediction and Top-N recommendation

The default ranking criterion for the creation of the top-N recommendation lists is to use the predicted values for each item to rank them. Predicted values are computed by Equation 5.8 for the case of UCF. Then, we use the same predicted rating values (after a simple ranking) also for the generation of the top-N recommendation lists. That is, we sort (in descending order) the items according to predicted rating values, and recommend the first N of them [1].

$$Predicted - Rating(u, i) = \frac{\sum_{v \in U} \mathrm{P}(u, v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in U} |\mathrm{P}(u, v)|}, \tag{5.8}$$

where, $r_{v,i}$ is rating given by v to i, and $\bar{r}_v$ is v's rating average.This ranking criterion, denoted as Highest Predicted Ratings (HPR) item recommendation is influenced by the good accuracy of prediction that existing related work reports through the Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). HPR opts for recommending the items that are more probable to receive a higher rating. Algorithm 1 describes the steps of the neighborhood optimization.

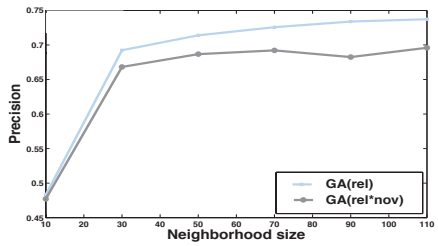
[1] If less than N items have positive ratings (i.e., not less than P_τ), then only less than N items are part of the recommendation list.

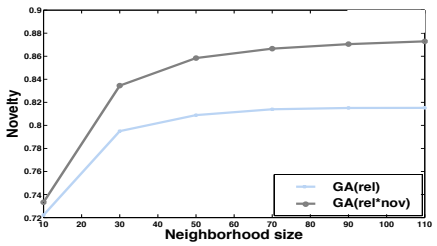
5.6 Sensitivity analysis of the proposed method

We have runned experiments to test the performance of the proposed method against state-of-the-art methods. We accomplished our experiments on Movielens 100k and Movielens by splitting the data into 80% for the training and 20% for the test. Furthermore, we employed precision, recall and novelty to validate the results on different neighborhood size, recommendation set lengths, and against baseline recommenders

We want to explore how the performance of the proposed method is affected in terms of providing more novel or more accurate recommendations. Our proposed fitness function, tries to optimize the neighborhood members, in a way, to provide both relevant and novel items by combining the aforementioned two criteria. To do this, we test the accuracy performance of two variations of the proposed method, denoted as GA(rel) and GA(rel*nov), when (i) we build a user's neighborhood with different numbers of k-nearest neighbors = 10, 30, 50, 70, 90, 110 and (ii) when we suggest a different number of top-N= 5, 10, 20 recommended items to each user. GA(rel) stands for the case where we run our genetic algorithm using only the relevance criterion for constructing neigborhoods, whereas GA(rel*nov) corresponds to the case where we use both the relevance and the novelty criteria for the construction of the target's user neigborhood.

Figure 5.1 and Figure 5.1 present how much precision and novelty we get, as we vary the neighborhood size. As it is shown in Figure 5.1(a) and Figure 5.1(a) by increasing the number of k nearest neigbhors, precision increases too, until it is stabilized around a neighborhood size of 110 users, whereas the performance of our method converges with a small drop off precision. Henchforth, we will run experiments using $k = 110$ which is the neighborhood size that gives the best precision. As expected, GA(rel) outperforms GA(rel*nov) in terms of precision, since using only the relevance criterion optimizes for more accurate recommendations.

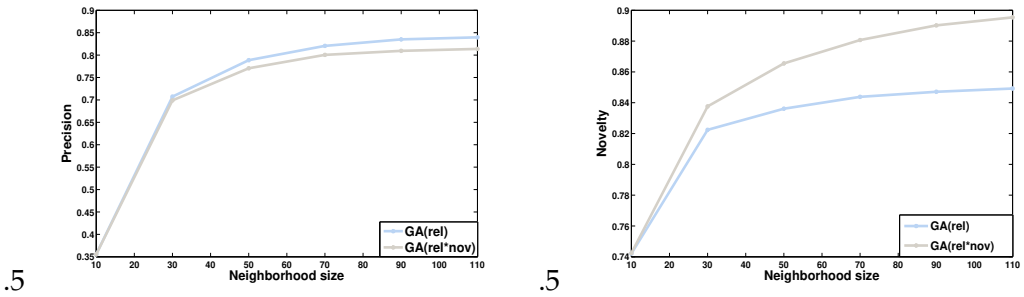




0.5 0.5

Figure 5.1: Analyzing the precision and novelty of (a) GA(rel) and (b) GA(rel*nov) model, versus the size of the neighborhood on Movielens 100K.

In contrast, as shown in Figure 5.1(b) and 5.1(b), GA(rel*nov) outperforms GA(rel) in terms of novelty for all different neighbourhood sizes, since it incorporates more novelty into the selected user neighbourhoods. Please notice by seeing together Fig-



.5 .5

Figure 5.1: Analyzing the precision and novelty of (a) GA(rel) and (b) GA(rel*nov) model, versus the size of the neighborhood on Movielens 1M.

ure 5.1(a) and Figure 5.1(b), also Figure 5.1(a) and Figure 5.1(b), that there is a well-known trade-off between precision and novelty. However, in our case we can attain more novel recommendations with minimal losses in terms of accuracy in order to attract users' attention.

Next, we want to test the accuracy, recall and novelty performance of both GA(rel) and GA(rel*nov), as we vary the top-N recommendation list size. Figure 5.1(a) and Figure 5.2(a) depict performances of GA(rel) on MovieLens 100K, and MovieLens 1M, as we vary the N number of recommended items, and Figure 5.1(b) and Figure 5.2(b) depict performances of GA(rel*nov)) on the datasets.

As expected, by increasing the number of N recommended movies, precision decreases rapidly for both GA(rel) and GA(rel*nov) models while recall increases. That is, precision is negatively correlated to the recommendation list size in contrary of recall which is positively correlated. However, we cannot say the same thing for novelty. That is, as we increase the number of recommended items, novelty either increases or decreases very slightly, whereas novelty difference between GA(rel) and GA(rel*nov) is more observable when the recommendation list size is smaller. The main reason can be that more items inside a recommendation list may bring more differentiation, but this is something that needs further investigation.

5.7 Performance comparison of the propsoed algorithm

5.7.1 Comparison with other Methods

In this Section, we compare the performance of GA(rev) and GA(rev*nov) with the following two comparison partners, which are representatives of different algorithmic families such as collaborative filtering, and clustering.

(i) User-based Collaborative Filtering (UCF) [Çoba 17]: Based on UCF, two users are considered similar, if they have rated similar items.

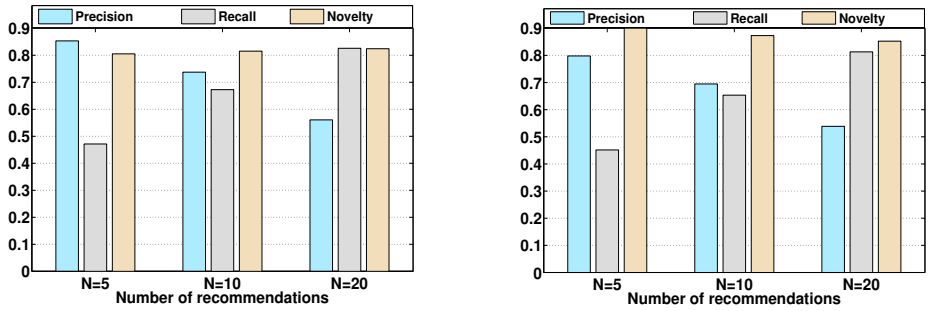


Figure 5.1: Analyzing the precision and novelty of (a) GA(rel) and (b) GA(rel*nov) model, versus the number of top-N recommendations on movielens 100K.

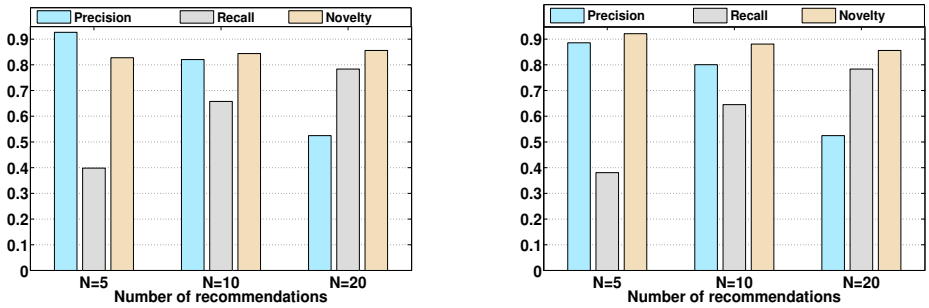


Figure 5.2: Analyzing the precision and novelty of (a) GA(rel) and (b) GA(rel*nov) model, versus the number of top-N recommendations on movielens 1M.

(ii) k-means clustering (k-means [Zahra 15]): k-means minimizes the intra-distance between the members of the same cluster and maximizes the inter-distance between centers of different cluster.

As it is shown in Figure 5.2((a),(b)), and Figure 5.2((a),(b)) GA(rel) outperforms the rest of algorithms in term of precision and recall. Please also notice that precision and recall of GA(rel*nov) are a bit worst than GA(rel), but still comparable to k-means and k-NN. Finally, Figure 5.2(c) and Figure 5.2(c) present that GA(rel) has close novelty to k-means and k-NN. As expected, GA(rel*nov) is far better than all three algorithms on both data sets, with minimal losses in terms of precision.

5.7.2 Comparison with other methods on cold users

In this section, we compare the performance of GA(rel) and GA(rel*nov) to k-means and k-NN methods on cold users. Thus, we seplited the data sets into three categories of users according to their coldness level. A cold user is defined by few ratings number which he gave during the interaction with the recommendation system []. We defined

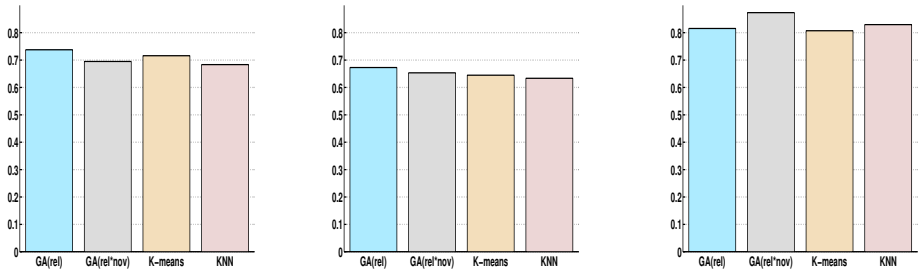


Figure 5.2: Comparing GA(rel) and GA(rel*nov) with other methods in terms of (a) precision, (b) recall, and (c) novelty on movielens 100K.

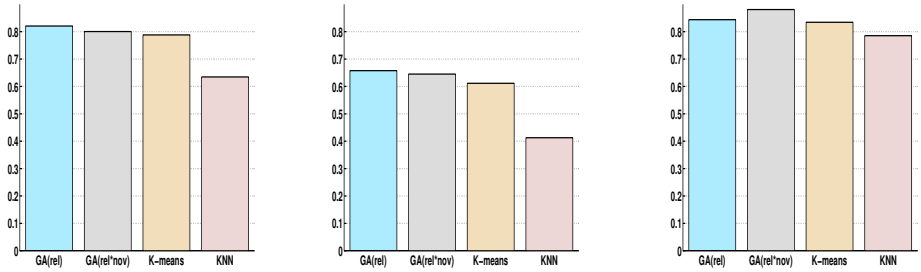


Figure 5.2: Comparing GA(rel) and GA(rel*nov) with other methods in terms of (a) precision, (b) recall, and (c) novelty on movielens 1M.

three categories of cold users, each user within these categories has a rating number in one of the next coldness ranges: CR1=[20 to 50], CR2=[50 to 100], and CR3=[20 to 100].

Results in Figures (9,10)(a), Figures (9,10)(b), and Figures (9,10)(c) demonstrate the performance of GA(rel) and GA(rel*nov) against *k*-means and *k*-NN in terms of Precision, Recall, and Novelty on CR1, CR2, and CR3 respectively.

As expected, Figures 9 and 10 demonstrate a negative correlation between coldness levels and relevancy metrics, wheras Precision and Recall on CR1 are worse than in CR2. However, these results become better on CR3 due the increased number of warm users involved in while validating the recommendations. GA(rel) kept giving best Precision and Recall on all coldness ranges, while GA(rel*nov) gave better novelty with a small lose in Precision, but it is still better than *k*-means and *k*-NN on CR3 for both data sets. Moreover, by comparing novelty results on CR1 and CR2 for both datasets, we observe a better performance on CR1, which proves that affecting more novel recommendations is easier for colder users, whereas GA(rel*nov) performed better than the rest of methods.

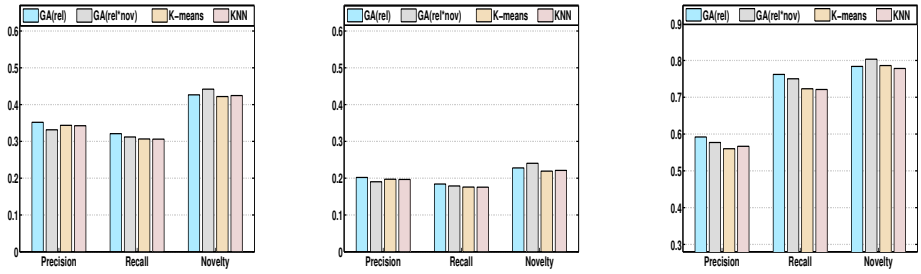


Figure 5.2: Comparing GA(rel) and GA(rel*nov) with other methods on cold users with movielens 100k.

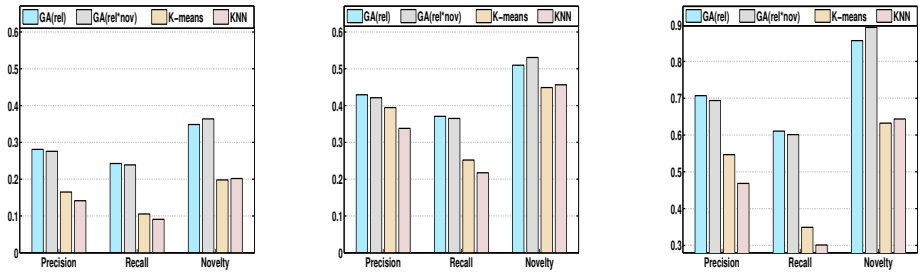


Figure 5.2: Comparing GA(rel) and GA(rel*nov) with other methods on cold users with movielens 100k.

In general, the obtained results showed better performances of GA in comparison to *k*-means and *k*-NN for both relevancy and novelty metrics due to its ability at personalizing a suitable neighborhood for each user individually and independently from the other users.

5.8 Conclusion

We have proposed an evolutionary-based approach to find the most suitable neighbors of a target user in terms of relevance and novelty. As postulated also in [Jannach 16] recommender systems research needs to go beyond optimizing purely on accuracy. Our proposed method employs a genetic algorithm to provide more novel recommendations with only a very small decrease in the accuracy of recommendations. Our experimental results show that we outperform baseline recommenders in terms of both novelty and precision. A limitation of this work is the single dataset and the limited set of comparison partners. However, in future work we want to extend our fitness function with additional data sources (e.g., diversity of items), explore alternative recommen-

dation paradigms for applying genetic algorithms and run experiments on more data sets.

When you reach the end of what you should know, you will be at the beginning of what you should sense

Kahlil Gibran

6 Conclusion

6.1 Introduction

Recommender systems are needed for the web to conserve users' time and energy. During the preparation of this thesis, efforts were made to improve the efficacy of these systems by focusing on two well-known and critical issues: the recommendations diversity, and the scalability. A proposed solution based on clustering was presented as a tool to solve at once these issues. The solution uses an evolutionary algorithm that puts all of the clustering configurations in one representation and optimizes the clustering quality progressively.

Even that the obtained results are comparable, and acceptable in comparison to previous contributions, the way in which these results have been achieved is questionable. In the next items, we discuss and evaluate the quality of the contributions made during this thesis. We also present some limitations, possible improvements, and future work.

6.2 Synthesis

Recommender systems benefited from a wide box of machine learning tools and covered a range of information sources. UCF is among the known recommendations techniques, whereas the recommendations are made based on two critical factors:

Selecting the neighborhood: As the collaborative filtering takes inspiration from the social life of humans, predicting the preferences of a given user is based on the

tastes of like-minded users. The similarity between two users can be calculated using similarity measures. This similarity is applied to different types of information such as ratings, demographic information, or social information.

- **Similarity measures:** By considering the users' ratings, we find that every two given users have a set of observed/consumed items. In fact, the similarity measures differ from each other in terms of their significance (e.g., closeness or correlation.), and their suitability for the recommendation context.
- **Size of the neighborhood:** This parameter has considerable importance, whereas a large neighborhood size may give better Precision in contrary to the smaller neighborhood. However, selecting among a wide set of users, a neighborhood is time taking, while a small neighborhood selection is more optimal. Balancing between the recommendation precision and optimizing time complexity is a must be.

Making prediction: The step after selecting a neighborhood is to make predictions. In this step, every neighbor shares his preferences with the target user, so a predicted rating is assigned to every item not seen by the target user. A prediction formula can be employed to evaluate how much a user may like an item.

- **Prediction formula:** The shared preferences values of the neighbors can be simply averaged to predict the preferences. More sophisticated methods consist of weighting the neighbors' preferences by their similarities to the target user. Prediction formulas are an open question that relies on, the quality of the recommendations.

Ranking the recommendations: Traditional recommenders have only targeted the relevancy of recommendations. However, the diversity and novelty of recommendations are recently taking more importance. To this end, re-ranking methods have appeared as an alternative to balance two conflicting objectives: the diversity/novelty of the recommendations, and the relevancy of recommendations.

As we look at the previous factors, we find that traditional UCF algorithms suffer some issues which we can summarize in the next:

Similarity is inefficient: The similarity measure takes only into consideration co-rated items. The profiles of two users can have 0 or more co-rated items. As the number of the co-rated items is small, the accuracy of the similarity is unreliable. As a solution, more data sources can be integrated, in particular social information.

Optimization of the neighborhood size is hard: The neighborhood size ranges from small sizes like 20 too much bigger sizes, which hardness the optimization of this parameter. For some users, few numbers of neighbors can lead to good accuracy, however for other ones' bigger number of neighbors is required.

Recommendations are redundant and inaccurate: Recommendation quality depends on many factors including the similarity measures as a critical keystone. In fact, using the similarity measures creates common profiles that unify many members with homogeneous experiences, which limits the diversity of the recommendations. However, using dissimilarity measures may give more diverse recommendations but drops down the relevancy. A recommender system must balance these two objectives.

Neighborhood identification is time taking in large users set: Neighbors selection takes a factorial time complexity, whereas for N user $N \times (N-1)$ similarity values must be calculated. After calculating the similarities between all pairs of users, most K close users should be selected. As K goes up as the time to accomplish the selection task goes longer.

6.3 Discussion

Genetic algorithms are among the most traditional evolutionary algorithms, which are known for their suitability for large optimization problems. Genetic algorithms allow to address more than one optimization problem at once and to explore an exponential search space more efficiently.

6.3.1 Issues:

Genetic algorithms have been proven to be useful for some problems in recommender systems. However, they still suffer some related issues:

Genetic algorithms are time taking:

What are the factors influencing the time complexity of GAs ? Genetic algorithms are inspired by the evolution theory, whereas many factors and parameters play a role to accomplish the optimization task. Parameters such as the initial population size, the initialization method, the size of the search space, the efficacy of the fitness function, and the mutations and crossover probabilities are all critical factors.

Why GAs' solutions are hard to code and decode? The encoded solution may be of a big size. In the case of binary representations, a simple transformation from binary to real values may become a very time taking task especially when the number of parameters is considerable.

Why evaluating a solution is a time taking? A fitness function in genetic algorithms should be fast calculated, however, in some cases, the optimization problem may be of high complexity. If we take a simple example as the density function employed in k-means algorithm, the similarities between the users can be

computed only one time then stored in memory which limits a bit the time complexity. However, when using more complex quality definitions like MAE, the fitness function must be recalculated every time we evaluate a solution ,which increases the time of evaluation.

Why is the behavior of a GA unpredictable? GA behaves according to the initially created population. In some cases, this population may be of small size in comparison to the complexity of the search space. In some other cases, the initial population may have a low diversity which may lead to a locally optimal solution. These cases should be taken into consideration while deploying a genetic algorithm.

The fitness function is hard to fix:

Why is it hard to find a mathematical modelization of the given problem ?
In contrary to supervised learning in the classification field, where individuals tickets are considered to supervise the classification learning, in genetic algorithms only the users' preferences are considered, while the quality metrics such as MAE, Diversity, or Precision are so time taking. The question which we need to answer is how to find a quality description of the neighborhoods without being in need to use the recommendation quality metrics. In other words, how to say that a neighborhood is beneficial based only on the users' ratings.

6.4 Future work and perspectives

As we presented previously, GA suffers two major problems because of time complexity. In the next items, we note some ideas which deserve a future investigation to alleviate this issue:

The clustering can be divided into sub-tasks, thus more questions can be raised:

- **How far is new computation paradigms such as map-reduce are efficient?** Map-reduce can be seen as an engineering solution that divides the processing task into sub-tasks. It has been deployed for k-NN algorithm and proved a significant performance. More ideas such as investigating using this paradigm for coding/decoding a chromosome, also computing similarity measures seem to be interesting experimental tracks.
- **Is parallelization a better choice?** In RS, some tasks can be done in parallel such as: computing the similarities, predicting the ratings for every user independently. Another research path is to deploy an optimization algorithm for every problem in parallel, then find a way to combine/coordinate the results of these algorithms. However, this solution needs experimentation to prove its efficacy face to other possibilities.

- **How can we define a fast computing fitness function in genetic-based optimization?** Fitness function specification is a critical choice in deploying a genetic algorithm. Avoiding hard computing functions can improve its efficacy significantly. A way to achieve this objective is: a) to avoid using direct quality indicators such as MAE. b) to use a small portion of test data while optimization. c) to look for a fitness function that depends on the initial data features and not the current state of the encoded solution (i.e., the complexity of the calculation should be independent of the encoded solution, so some calculations will not be recalculated more than one time.)
- **Can we combine different optimization algorithms?** Optimization algorithms differ from each other on many bases such as a)their suitability for specific problems (e.g., discrete or continuous values.), b) the manner in which the initial solution set evolves. In fact, optimization algorithms were combined together to improve their efficacy. We consider as well this possibility in our work as an enhancement opportunity. The general island model is a recent technique that already put the theory and the application of this new optimization technique.

More questions can be raised to improve the recommendation quality such as **how can we benefit from the variety of information sources?** In fact, hybrid recommender systems have been proven to be more efficient than the rest of the individual recommenders, whereas GA was used to weigh the contribution of each recommender. Based on our previous discussion about the insufficient information problem, the idea of combining more data sources seems to be interesting to improve the clustering quality. In such a case, GA can be used as a feature extraction tool or a linear combination function that optimizes the similarity values or the prediction weights.

6.5 Conclusion

We have discussed in this chapter recommender system issues. We presented our contributions followed by an analysis to manifest their advantages and disadvantage. We believe that there is no border to enhance, develop, and explore better techniques to solve the recommendation issues. While the main contributions of this thesis considered movies recommendations, additional interesting recommendations applications are still attractive such as e-learning, news recommendation, and healthcare.

Bibliography

[Abdollahpouri 17] Himan Abdollahpouri, Robin Burke & Bamshad Mobasher. *Controlling Popularity Bias in Learning-to-Rank Recommendation*. In Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys 17, pages 42–46, 2017. 55

[Acilar 09] A. Merve Acilar & Ahmet Arslan. *A collaborative filtering method based on artificial immune network*. Expert Systems with Applications, vol. 36, no. 4, pages 8324 – 8332, 2009. 46, 56

[Adomavicius 05a] G. Adomavicius & A. Tuzhilin. *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, vol. 17, no. 6, pages 734–749, jun 2005. 10

[Adomavicius 05b] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen & Alexander Tuzhilin. *Incorporating contextual information in recommender systems using a multidimensional approach*. ACM Transactions on Information Systems (TOIS), vol. 23, no. 1, pages 103–145, 2005. 13

[Adomavicius 10] Gediminas Adomavicius & Alexander Tuzhilin. *Context-Aware Recommender Systems*. In Recommender Systems Handbook, pages 217–253. Springer US, October 2010. 27

[Aggarwal 99] Charu C Aggarwal, Joel L Wolf, Kun-Lung Wu & Philip S Yu. *Horting hatches an egg: A new graph-theoretic approach to collaborative filtering*. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 201–212. ACM, 1999. 43

[Al Mamunur Rashid 06] Shyong K Lam Al Mamunur Rashid, George Karypis & John Riedl. *ClustKNN: a highly scalable hybrid model-& memory-based CF algorithm*. Proceeding of webKDD, vol. 2006, 2006. 15

[Alam 12] Shafiq Alam, Gillian Dobbie, Patricia Riddle & Yun Sing Koh. *Hierarchical PSO clustering based recommender system*. In

2012 IEEE Congress on Evolutionary Computation, pages 1–8. IEEE, 2012. 29

[Alhijawi 16a] B. Alhijawi & Y. Kilani. *Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems*. In 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), pages 1–6, jun 2016. 80

[Alhijawi 16b] Bushra Alhijawi & Yousef Kilani. *Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems*. In 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), pages 1–6, 06 2016. 61

[Alsarhan 18] Ayoub Alsarhan, Bushra Alhijawi & Yousef Kilani. *Using Artificial Intelligence Techniques in Collaborative Filtering Recommender Systems: Survey*. International Journal of Advanced Intelligence Paradigms, vol. 10, no. 1, page p1, 2018. 29

[Aytekin 13] Tevfik Aytekin & Mahmut Özge Karakaya. *Clustering-based diversity improvement in top-N recommendation*. Journal of Intelligent Information Systems, vol. 42, no. 1, pages 1–18, jun 2013. 55

[Bag 19] Sujoy Bag, Abhijeet Ghadge & Manoj Kumar Tiwari. *An integrated recommender system for improved accuracy and aggregate diversity*. Computers & Industrial Engineering, vol. 130, pages 187–197, April 2019. 56

[Bedi 12] Punam Bedi & Ravish Sharma. *Trust based recommender system using ant colony for trust computation*. Expert Systems with Applications, vol. 39, no. 1, pages 1183–1190, January 2012. 28

[Bedi 15] Punam Bedi & Ravish Sharma. *Ant-based friends recommendation in social tagging systems*. International Journal of Swarm Intelligence, vol. 1, page p321, 01 2015. 28

[Bellogín 13] Alejandro Bellogín, Pablo Castells & Iván Cantador. *Improving memory-based collaborative filtering by neighbour selection based on user preference overlap*. In Proceedings of the 10th conference on open research areas in information retrieval, pages 145–148, 2013. 39

[Bellogín 14] Alejandro Bellogín, Pablo Castells & Iván Cantador. *Neighbor Selection and Weighting in User-Based Collaborative Filtering*. ACM Transactions on the Web, vol. 8, no. 2, pages 1–30, March 2014. 47, 102

[Bennett 07] James Bennett, Stan Lanning*et al. The netflix prize*. In Proceedings of KDD cup and workshop, volume 2007, page 35. Citeseer, 2007. 9, 24

[Berbague 18a] ChemsEddine Berbague, Nour El islem Karabadji & Hassina Seridi. *Enhancing the Sales Diversity Using a Two-Stage Improved KNN Algorithm*. In International Symposium on Modelling and Implementation of Complex Systems, pages 193–203. Springer, 2018. 71

[Berbague 18b] ChemsEddine Berbague, Nour El-Islam Karabadji & Hassina Seridi. *Recommendation diversification using a weighted similarity measure in user based collaborative filtering*. In Proceedings of 2018 International Symposium on Programming and Systems (ISPS), pages 1–6, April 2018. 56

[Berbague 18c] ChemsEddine Berbague, Nour El Islem Karabadji & Hassina Seridi. *An Evolutionary Scheme for Improving Recommender System Using Clustering*. In CIIA, volume 522 of *IFIP Advances in Information and Communication Technology*, pages 290–301. Springer, 2018. 77

[Berners-Lee 01] Tim Berners-Lee & Mark Fischetti. Weaving the web: The original design and ultimate destiny of the world wide web by its inventor. DIANE Publishing Company, 2001. 9

[Billsus 98] Daniel Billsus & Michael J Pazzani. *Learning Collaborative Information Filters.* In Icml, volume 98, pages 46–54, 1998. 45

[Birtolo 13] Cosimo Birtolo & Davide Ronca. *Advances in Clustering Collaborative Filtering by means of Fuzzy C-means and trust*. Expert Systems with Applications, vol. 40, no. 17, pages 6997–7009, dec 2013. 85

[Bobadilla 11a] Jesus Bobadilla, Antonio Hernando, Fernando Ortega & Jesus Bernal. *A framework for collaborative filtering recommender systems*. Expert Systems with Applications, vol. 38, no. 12, pages 14609–14623, November 2011. 43

[Bobadilla 11b] Jesus Bobadilla, Fernando Ortega, Antonio Hernando & Javier Alcalá. *Improving collaborative filtering recommender system results and performance using genetic algorithms*.

Knowledge-based systems, vol. 24, no. 8, pages 1310–1316, 2011. 28, 29

[Bobadilla 13] J. Bobadilla, F. Ortega, A. Hernando & A. Gutiérrez. *Recommender systems survey*. Knowledge-Based Systems, vol. 46, pages 109–132, 2013. ix, 12, 13, 20, 27, 35, 99

[Boim 11] Rubi Boim, Tova Milo & Slava Novgorodov. *Diversification and refinement in collaborative filtering recommender*. In Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM 11, pages 739–744. ACM Press, 2011. 55

[Bokde 15] Dheeraj Bokde, Sheetal Girase & Debajyoti Mukhopadhyay. *Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey*. Procedia Computer Science, vol. 49, pages 136–146, 2015. 14, 36, 54

[Bradley 01] Keith Bradley & Barry Smyth. *Improving recommendation diversity*. In Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland, pages 85–94. Citeseer, 2001. 11

[Breese 98] John S Breese, David Heckerman & Carl Kadie. *Empirical analysis of predictive algorithms for collaborative filtering*. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pages 43–52. Morgan Kaufmann Publishers Inc., 1998. 49

[Burke 02a] Robin Burke. User Modeling and User-Adapted Interaction, vol. 12, no. 4, pages 331–370, 2002. 16

[Burke 02b] Robin Burke. *Hybrid recommender systems: Survey and experiments*. User modeling and user-adapted interaction, vol. 12, no. 4, pages 331–370, 2002. 14

[Burke 07] Robin Burke. *Hybrid web recommender systems*. In The adaptive web, pages 377–408. Springer, 2007. 13

[Cacheda 11] Fidel Cacheda, Víctor Carneiro, Diego Fernández & Vreixo Formoso. *Comparison of collaborative filtering algorithms*. ACM Transactions on the Web, vol. 5, no. 1, pages 1–33, feb 2011. 18

[CAI 20] *A hybrid recommendation system with many-objective evolutionary algorithm*. Expert Systems with Applications, vol. 159, page 113648, 2020. 56

[Carbonell 98] Jaime Carbonell & Jade Goldstein. *The use of MMR, diversity-based reranking for reordering documents and producing summaries*. In Proceedings of the 21st annual international ACM conference on Research and development in information retrieval - SIGIR 98, pages 335–336. ACM Press, 1998. 53

[Castells 11] Pablo Castells, SaÃºl Vargas & Jun Wang. *Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance*. Proceedings of International Workshop on Diversity in Document Retrieval (DDR), pages 29–36, 01 2011. 81

[Celma 10] O. Celma. Music Recommendation and Discovery in the Long Tail. Springer, 2010. 25

[Chatzicharalampous 15] Evangelos Chatzicharalampous, Zigkolis Christos & Athena Vakali. *Exploriometer: Leveraging Personality Traits for Coverage and Diversity Aware Recommendations*. In Proceedings of the 24th International Conference on World Wide Web - WWW 15 Companion, pages 1463–1468. ACM Press, 2015. 54, 56, 79

[Chen 13] Chaochao Chen, Jing Zeng, Xiaolin Zheng & Deren Chen. *Recommender system based on social trust relationships*. In 2013 IEEE 10th International Conference on e-Business Engineering, pages 32–37. IEEE, 2013. 38

[Cheng 16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir*et al. Wide & deep learning for recommender systems*. In Proceedings of the 1st workshop on deep learning for recommender systems, pages 7–10. ACM, 2016. 14

[Chifu 15] Viorica R. Chifu, Ioan Salomie, Emil Şt. Chifu, Cristina Bianca Pop, Dan Valea, Madalina Lupu & Marcel Antal. *Hybrid Invasive Weed Optimization Method for Generating Healthy Meals*. In Soft Computing Applications, pages 335–351. Springer International Publishing, November 2015. 28

[Cho 02] Yoon Ho Cho, Jae Kyeong Kim & Soung Hie Kim. *A personalized recommender system based on web usage mining and decision tree induction*. Expert systems with Applications, vol. 23, no. 3, pages 329–342, 2002. 14

[Christakopoulou 16] Evangelia Christakopoulou & George Karypis. *Local Item-Item Models For Top-N Recommendation*. In Proceedings of the

10th ACM Conference on Recommender Systems - RecSys 16, pages 67–74. ACM Press, 2016. 54

[Çoba 17] Ludovik Çoba, Panagiotis Symeonidis & Markus Zanker. *Reproducing and Prototyping Recommender Systems in R*. In Proceedings of the 8th Italian Information Retrieval Workshop, Lugano, Switzerland., 2017. 106

[Coba 18] Ludovik Coba, Panagiotis Symeonidis & Markus Zanker. *Novelty-Aware Matrix Factorization Based on Itemsâ Popularity*. In International Conference of the Italian Association for Artificial Intelligence, pages 516–527. Springer, 2018. 55

[Croft 10] W Bruce Croft, Donald Metzler & Trevor Strohman. Search engines: Information retrieval in practice, volume 520. Addison-Wesley Reading, 2010. 9

[Dai 09] Yae Dai, HongWu Ye & SongJie Gong. *Personalized recommendation algorithm using user demography information*. In 2009 Second International Workshop on Knowledge Discovery and Data Mining, pages 100–103. IEEE, 2009. 15

[Dakhel 11] Gilda Moradi Dakhel & Mehregan Mahdavi. *A new collaborative filtering algorithm using K-means clustering and neighbors' voting*. In 2011 11th International Conference on Hybrid Intelligent Systems (HIS), pages 179–184. IEEE, dec 2011. 15, 55

[de Aguiar Neto 20] Fernando S. de Aguiar Neto, Arthur F. da Costa, Marcelo G. Manzato & Ricardo J.G.B. Campello. *Pre-processing approaches for collaborative filtering based on hierarchical clustering*. Information Sciences, vol. 534, pages 172–191, 2020. 85

[Deb 95] Kalyanmoy Deb & Ram Bhushan Agrawal. *Simulated binary crossover for continuous search space*. Complex systems, vol. 9, no. 2, pages 115–148, 1995. 86

[Deb 96] Kalyanmoy Deb & Mayank Goyal. *A Combined Genetic Adaptive Search (GeneAS) for Engineering Design*. Computer Science and Informatics, vol. 26, pages 30–45, 1996. 86

[Demovic 13] Lubos Demovic, Eduard Fritscher, Jakub Kriz, Ondrej Kuzmik, Ondrej Proksa, Diana Vandlikova, Dusan Zelenik & Maria Bielikova. *Movie recommendation based on graph traversal algorithms*. In 2013 24th International Workshop on Database and Expert Systems Applications, pages 152–156. IEEE, 2013. 18

[Deshpande 04] Mukund Deshpande & George Karypis. *Item-based top-n recommendation algorithms*. ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, pages 143–177, 2004. 45

[Desrosiers 11] Christian Desrosiers & George Karypis. *A comprehensive survey of neighborhood-based recommendation methods*. In Recommender systems handbook, pages 107–144. Springer, October 2011. 14, 40, 42, 47

[Ekstrand 11] Michael D Ekstrand, John T Riedl, Joseph A Konstan*et al. Collaborative filtering recommender systems*. Foundations and Trends® in Human–Computer Interaction, vol. 4, no. 2, pages 81–173, 2011. 14

[Fouss 06] Francois Fouss, Luh Yen, Alain Pirotte & Marco Saerens. *An experimental investigation of graph kernels on a collaborative recommendation task*. In Sixth International Conference on Data Mining (ICDM'06), pages 863–868. IEEE, 2006. 43

[Fouss 07] Francois Fouss, Alain Pirotte, Jean-Michel Renders & Marco Saerens. *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation*. IEEE Transactions on knowledge and data engineering, vol. 19, no. 3, pages 355–369, 2007. 43

[Georgiou 10] Olga Georgiou & Nicolas Tsapatsoulis. *Improving the scalability of recommender systems by clustering using genetic algorithms*. In International conference on artificial neural networks, pages 442–449. Springer, 2010. 15

[Gibbons 90] Jean D Gibbons & Maurice Kendall. *Rank correlation methods*. Edward Arnold, 1990. 46

[Godhani 17] Goral Godhani & Maulik Dhamecha. *A study on movie recommendation system using parallel MapReduce technology*. International Journal of Engineering Development and Research, vol. 5, no. 1, pages 7683–7692, 2017. 51

[Gogna 17] Anupriya Gogna & Angshul Majumdar. *DiABlO: Optimization based design for improving diversity in recommender system*. Information Sciences, vol. 378, pages 59–74, February 2017. 55

[Gohari 17] Faezeh Sadat Gohari, Hassan Haghighi & Fereidoon Shams Aliee. *A semantic-enhanced trust based recommender system using ant colony optimization*. Applied Intelligence, vol. 46, no. 2, pages 328–364, 2017. 29

[Golub 71] Gene H Golub & Christian Reinsch. *Singular value decomposition and least squares solutions*. In Linear Algebra, pages 134–151. Springer, 1971. 36

[Guimarães 13] Adolfo P. Guimarães, Thales F. Costa, Anísio Lacerda, Gisele L. Pappa & Nivio Ziviani. *GUARD: A Genetic Unified Approach for Recommendation*. JIDM, vol. 4, no. 3, pages 295–310, 2013. 61

[Gunawardana 09] Asela Gunawardana & Guy Shani. *A survey of accuracy evaluation metrics of recommendation tasks*. Journal of Machine Learning Research, vol. 10, no. Dec, pages 2935–2962, 2009. 20

[Gupta 17] Aditya Gupta & Kunal Gusain. *Selection of Similarity Function for Context-Aware Recommendation Systems*. In Computational Intelligence in Data Mining, pages 803–811. Springer, 2017. 28

[Harper 16] F Maxwell Harper & Joseph A Konstan. *The movielens datasets: History and context*. Acm transactions on interactive intelligent systems (tiis), vol. 5, no. 4, page 19, 2016. 24

[He 10] Jianming He & Wesley W Chu. *A social network-based recommender system (SNRS)*. In Data mining for social network data, pages 47–74. Springer, 2010. 13

[Herlocker 99] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers & John Riedl. *An algorithmic framework for performing collaborative filtering*. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 99. ACM Press, 1999. 49

[Herlocker 02] Jon Herlocker, Joseph A Konstan & John Riedl. *An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms*. Information retrieval, vol. 5, no. 4, pages 287–310, 2002. 46

[Herlocker 04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen & John T. Riedl. *Evaluating collaborative filtering recommender systems*. ACM TRANSACTIONS ON INFORMATION SYSTEMS, vol. 22, pages 5–53, 2004. 23, 24

[HewaNadungodage 17] Chandima HewaNadungodage, Yuni Xia & John Jaehwan Lee. *A GPU-oriented online recommendation algorithm for efficient processing of time-varying continuous data streams*. Knowl-

edge and Information Systems, vol. 53, no. 3, pages 637–670, 2017. 51

[Hidasi 16] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou & Domonkos Tikk. *Parallel recurrent neural network architectures for feature-rich session-based recommendations*. In Proceedings of the 10th ACM conference on recommender systems, pages 241–248. ACM, 2016. 51

[Honda 01] Katsuhiro Honda, Nobukazu Sugiura, Hidetomo Ichihashi & Shoichi Araki. *Collaborative Filtering Using Principal Component Analysis and Fuzzy Clustering*. In Web Intelligence: Research and Development, pages 394–402. Springer Berlin Heidelberg, 2001. 44

[Huang 04] Zan Huang, Hsinchun Chen & Daniel Zeng. *Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering*. ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, pages 116–142, 2004. 43

[Isinkaye 15] FO Isinkaye, YO Folajimi & BA Ojokoh. *Recommendation systems: Principles, methods and evaluation*. Egyptian Informatics Journal, vol. 16, no. 3, pages 261–273, 2015. 10

[Jamali 09] Mohsen Jamali & Martin Ester. *TrustWalker : a random walk model for combining trust-based and item-based recommendation*. pages 397–406, 06 2009. 18

[Jamali 10] Mohsen Jamali & Martin Ester. *A matrix factorization technique with trust propagation for recommendation in social networks*. In Proceedings of the fourth ACM conference on Recommender systems, pages 135–142. ACM, 2010. 38

[Jannach 10] Dietmar Jannach, Markus Zanker, Alexander Felfernig & Gerhard Friedrich. Recommender systems: an introduction. Cambridge University Press, 2010. 99

[Jannach 16] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin & Markus Zanker. *Recommender systemsâbeyond matrix completion*. Communications of the ACM, vol. 59, no. 11, pages 94–102, 2016. 109

[Jawaheer 14] Gawesh Jawaheer, Peter Weller & Patty Kostkova. *Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback*. ACM Transactions on Interactive Intelligent Systems (TiiS), vol. 4, no. 2, page 8, 2014. 13

[Jin 04] Rong Jin, Joyce Y Chai & Luo Si. *An automatic weighting scheme for collaborative filtering*. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pages 337–344. ACM, 2004. 47

[Ju 13] Chunhua Ju & Chonghuan Xu. *A new collaborative recommendation approach based on users clustering using artificial bee colony algorithm*. The Scientific World Journal, vol. 2013, pages 1–9, 2013. 56

[Kagita 14] Venkateswara Rao Kagita, Arun K Pujari, Vineet Padmanabhan*et al*. *Collaborative filtering by PSO-based MMMF*. In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 569–574. IEEE, 2014. 29

[Kaleroun 14] Abhishek Kaleroun & Shalini Batra. *Collaborating trust and item-prediction with ant colony for recommendation*. In 2014 Seventh International Conference on Contemporary Computing (IC3). IEEE, August 2014. 28

[Kaminskas 17] Marius Kaminskas & Derek Bridge. *Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems*. ACM Transactions on Interactive Intelligent Systems (TiiS), vol. 7, no. 1, page 2, 2017. 11

[Karabadji 18] Nour El Islem Karabadji, Samia Beldjoudi, Hassina Seridi, Sabeur Aridhi & Wajdi Dhifli. *Improving memory-based user collaborative filtering with evolutionary multi-objective optimization*. Expert Systems with Applications, vol. 98, pages 153 – 165, 2018. 44, 56, 61, 62, 77, 86, 100, 101

[Katarya 16] Rahul Katarya & Om Prakash Verma. *A collaborative recommender system enhanced with particle swarm optimization technique*. Multimedia Tools and Applications, vol. 75, no. 15, pages 9225–9239, 2016. 29, 44

[Katarya 17] Rahul Katarya & Om Prakash Verma. *Efficient music recommender system using context graph and particle swarm*. Multimedia Tools and Applications, vol. 77, no. 2, pages 2673–2687, February 2017. 28

[Katarya 18] Rahul Katarya. *Movie recommender system with metaheuristic artificial bee*. Neural Computing and Applications, vol. 30, no. 6, pages 1983–1990, 2018. 29

[Kaya 18] Mesut Kaya & Derek Bridge. *Accurate and diverse recommendations using item-based subprofiles*. In The Thirty-First International Flairs Conference, pages 462–467, 2018. 53

[Kim 08] Kyoung-jae Kim & Hyunchul Ahn. *A recommender system using GA K-means clustering in an online shopping market*. Expert systems with applications, vol. 34, no. 2, pages 1200–1209, 2008. 28, 29, 56

[Kim 17] Donghyun Kim, Chanyoung Park, Jinoh Oh & Hwanjo Yu. *Deep hybrid recommender systems via exploiting document context and statistics of items*. Information Sciences, vol. 417, pages 72–87, 2017. 16

[Koohi 16] Hamidreza Koohi & Kourosh Kiani. *User based collaborative filtering using fuzzy C-means*. Measurement, vol. 91, pages 134–139, 2016. 19

[Koren 10] Yehuda Koren. *Collaborative filtering with temporal dynamics*. Communications of the ACM, vol. 53, no. 4, page p89, April 2010. 37

[Kosala 00] Raymond Kosala & Hendrik Blockeel. *Web mining research: A survey*. ACM Sigkdd Explorations Newsletter, vol. 2, no. 1, pages 1–15, 2000. 9

[Krulwich 97] Bruce Krulwich. *Lifestyle finder: Intelligent user profiling using large-scale demographic data*. AI magazine, vol. 18, no. 2, pages 37–37, 1997. 19

[Kumar 17] Akshi Kumar, Nitin Sachdeva & Archit Garg. *Analysis of GA Optimized ANN for Proactive Context Aware Recommender System*. In International Conference on Health Information Science, pages 92–102. Springer, 2017. 29

[Lam 08] Xuan Nhat Lam, Thuc Vu, Trong Duc Le & Anh Duc Duong. *Addressing cold-start problem in recommendation systems*. In Proceedings of the 2nd international conference on Ubiquitous information management and communication, pages 208–211. ACM, 2008. 25

[Li 03] Qing Li & Byeong Man Kim. *Clustering approach for hybrid recommender system*. In Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003), pages 33–38. IEEE, 2003. 15

[Lika 14] Blerina Lika, Kostas Kolomvatsos & Stathes Hadjiefthymiades. *Facing the cold start problem in recommender systems*. Expert Systems with Applications, vol. 41, no. 4, pages 2065–2073, 2014. 25

[Logesh 19a] R Logesh & V Subramaniyaswamy. *Exploring hybrid recommender systems for personalized travel applications*. In Cognitive informatics and soft computing, pages 535–544. Springer, 2019. 16

[Logesh 19b] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, Xiao-Zhi Gao & Gai-Ge Wang. *Hybrid bio-inspired user clustering for the generation of diversified recommendations*. Neural Computing and Applications, vol. 32, no. 7, pages 2487–2506, March 2019. 56

[Lops 11] Pasquale Lops, Marco De Gemmis & Giovanni Semeraro. *Content-based recommender systems: State of the art and trends*. In Recommender systems handbook, pages 73–105. Springer, 2011. 13, 14

[Ma 08] Hao Ma, Haixuan Yang, Michael R Lyu & Irwin King. *Sorec: social recommendation using probabilistic matrix factorization*. In Proceedings of the 17th ACM conference on Information and knowledge management, pages 931–940. ACM, 2008. 38

[Mehta 15] Siddharth J. Mehta & Jinkal Javia. *Threshold based KNN for fast and more accurate recommendations*. In 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS). IEEE, July 2015. 43

[Mehta 17] Rachana Mehta & Keyur Rana. *A review on matrix factorization techniques in recommender systems*. In 2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA), pages 269–274. IEEE, 2017. 51

[Mnih 08] Andriy Mnih & Ruslan R Salakhutdinov. *Probabilistic matrix factorization*. In Advances in neural information processing systems, pages 1257–1264, 2008. 38

[Nilashi 16a] Mehrbakhsh Nilashi, Mohammad Dalvi Esfahani, Morteza Zamani Roudbaraki, T Ramayah & Othman Ibrahim. *A multi-criteria collaborative filtering recommender system using clustering and regression techniques*. Journal of Soft Computing and Decision Support Systems, vol. 3, no. 5, pages 24–30, 2016. 26

[Nilashi 16b] Mehrbakhsh Nilashi, Maryam Salahshour, Othman Ibrahim, Abbas Mardani, Mohammad Dalvi Esfahani & Norhayati Zakuan. *A new method for collaborative filtering recommender systems: the case of yahoo! movies and tripadvisor datasets*. Journal of Soft Computing and Decision Support Systems, vol. 3, no. 5, pages 44–46, 2016. 37

[Nilashi 18] Mehrbakhsh Nilashi, Othman Ibrahim & Karamollah Bagherifard. *A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques*. Expert Systems with Applications, vol. 92, pages 507–520, 2018. 26

[Ning 11] Xia Ning & George Karypis. *SLIM: Sparse Linear Methods for Top-N Recommender Systems*. In Proceedings of the 11th International Conference on Data Mining, pages 497–506, 12 2011. 54

[Panigrahi 16] Sasmita Panigrahi, Rakesh Ku Lenka & Ananya Stitipragyan. *A hybrid distributed collaborative filtering recommender engine using apache spark*. Procedia Computer Science, vol. 83, pages 1000–1006, 2016. 51

[Paradarami 17] Tulasi K Paradarami, Nathaniel D Bastian & Jennifer L Wightman. *A hybrid recommender system using artificial neural networks*. Expert Systems with Applications, vol. 83, pages 300–313, oct 2017. 16, 29

[Parambath 16] Shameem A. Puthiya Parambath, Nicolas Usunier & Yves Grandvalet. *A Coverage-Based Approach to Recommendation Diversity On Similarity Graph*. In Proceedings of the 10th ACM Conference on Recommender Systems - RecSys 16, pages 15–22. ACM Press, 2016. 53

[Park 07] Moon-Hee Park, Jin-Hyuk Hong & Sung-Bae Cho. *Location-based recommendation system using bayesian userâs preference model in mobile devices*. In International conference on ubiquitous intelligence and computing, pages 1130–1139. Springer, 2007. 14

[Park 08] Yoon-Joo Park & Alexander Tuzhilin. *The long tail of recommender systems and how to leverage it*. In Proceedings of the 2008 ACM conference on Recommender systems, pages 11–18. ACM, 2008. 27, 51, 79

[Parvin 19] Hashem Parvin, Parham Moradi & Shahrokh Esmaeili. *TCFACO: Trust-aware collaborative filtering method based on ant colony optimization*. Expert Systems with Applications., vol. 118, pages 152–168, sep 2019. 80

[Patra 19] Sukanya Patra & Boudhayan Ganguly. *Improvising Singular Value Decomposition by KNN for Use in Movie Recommender Systems*. Journal of Operations and Strategic Planning, vol. 2, no. 1, pages 22–34, 2019. 37

[Pereira 15] Andre Luiz Vizine Pereira & Eduardo Raul Hruschka. *Simultaneous co-clustering and learning to address the cold start problem in recommender systems*. Knowledge-Based Systems, vol. 82, pages 11–19, jul 2015. 55

[Peška 19] Ladislav Peška, Tsegaye Misikir Tashu & Tomáš Horváth. *Swarm intelligence techniques in recommender systems - A review of recent research*. Swarm and Evolutionary Computation, vol. 48, pages 201–219, August 2019. 28, 29, 30, 31

[Porcel 12] C. Porcel, A. Tejeda-Lorente, M.A. Martínez & E. Herrera-Viedma. *A hybrid recommender system for the selective dissemination of research resources in a Technology Transfer Office*. Information Sciences, vol. 184, no. 1, pages 1–19, February 2012. 19

[Rad 07] Hoda Sepehri Rad & Caro Lucas. *A recommender system based on invasive weed optimization algorithm*. In 2007 IEEE Congress on Evolutionary Computation. IEEE, September 2007. 28

[Rahman 14] Md.Anisur Rahman & Md.Zahidul Islam. *A hybrid clustering technique combining a novel genetic algorithm with K-Means*. Knowledge-Based Systems, vol. 71, pages 345–365, 2014. 44

[Raja 18] Kumar Raja & S. Pushpa. *Novelty-driven recommendation by using integrated matrix factorization and temporal-aware clustering optimization*. International Journal of Communication Systems, page e3851, November 2018. 55

[Resnick 94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom & John Riedl. *GroupLens: an open architecture for collaborative filtering of netnews*. In Proceedings of the 1994 ACM conference on Computer supported cooperative work, pages 175–186. ACM, 1994. 49

[Ribeiro 12] Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso & Nivio Ziviani. *Pareto-efficient hybridization for multi-objective recommender systems*. In Proceedings of the sixth ACM conference on Recommender systems - RecSys 12, page p. ACM Press, 2012. 61

[Ribeiro 14] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda & Adriano Veloso. *Multiobjective Pareto-Efficient Approaches for Recommender Systems*. ACM Transactions on Intelligent Systems and Technology, vol. 5, no. 4, pages 1–20, dec 2014. 56

[Ricci 15] Francesco Ricci, Lior Rokach & Bracha Shapira. *Recommender Systems: Introduction and Challenges*. In Recommender Systems Handbook, pages 1–34. Springer US, 2015. 10, 11

[Rozie 14] Andri Fachrur Rozie & Hoh In. *Swarm collaborative filtering through fish school search*. International Journal of Software Engineering and its Applications, vol. 8, no. 3, pages 251–254, 2014. 30

[Safoury 13] Laila Safoury & Akram Salah. *Exploiting user demographic attributes for solving cold-start problem in recommender system*. Lecture Notes on Software Engineering, vol. 1, no. 3, pages 303–307, 2013. 15

[Salman 16] K. H. Salman, Arun K. Pujari, Vikas Kumar & Sowmini Devi Veeramachaneni. *Combining Swarm with Gradient Search for Maximum Margin Matrix Factorization*. In PRICAI 2016: Trends in Artificial Intelligence, pages 167–179. Springer International Publishing, 2016. 29

[Salter 06] James Salter & Nick Antonopoulos. *CinemaScreen recommender agent: combining collaborative and content-based filtering*. IEEE Intelligent Systems, vol. 21, no. 1, pages 35–41, 2006. 19

[Santos 10] Rodrygo L.T. Santos, Craig Macdonald & Iadh Ounis. *Exploiting query reformulations for web search result diversification*. In Proceedings of the 19th international conference on World wide web - WWW 10, pages 881–890. ACM Press, 2010. 53

[Sarwar 00] Badrul Sarwar, George Karypis, Joseph Konstan & John Riedl. *Application of dimensionality reduction in recommender system-a case study*. Rapport technique, Minnesota Univ Minneapolis Dept of Computer Science, 2000. 26

[Sarwar 01] Badrul Sarwar, George Karypis, Joseph Konstan & John Riedl. *Item-based collaborative filtering recommendation algorithms*. In Proceedings of the 10th international conference on World Wide Web, pages 285–295, 2001. 39

[Sarwar 02a] Badrul Sarwar, George Karypis, Joseph Konstan & John Riedl. *Incremental singular value decomposition algorithms for highly scalable recommender systems*. In Fifth international conference on computer and information science, volume 27, page p28. Citeseer, 2002. 26

[Sarwar 02b] Badrul M. Sarwar, George Karypis, Joseph Konstan & John Reidl. *Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering*. In Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT), volume 1, pages 291–324, 2002. 55

[Sarwar 02c] Badrul M Sarwar, George Karypis, Joseph Konstan & John Riedl. *Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering*. In Proceedings of the fifth international conference on computer and information technology, volume 1, pages 291–324, 2002. 26

[Schafer 07] J Ben Schafer, Dan Frankowski, Jon Herlocker & Shilad Sen. *Collaborative filtering recommender systems*. In The adaptive web, pages 291–324. Springer, 2007. 35

[Shambour 16] Qusai Shambour, Mouath Hourani & Salam Fraihat. *An item-based multi-criteria collaborative filtering algorithm for personalized recommender systems*. International Journal of Advanced Computer Science and Applications, vol. 7, no. 8, pages 274–279, 2016. 19

[Shani 11] Guy Shani & Asela Gunawardana. *Evaluating recommendation systems*. In Recommender systems handbook, pages 257–297. Springer, 2011. 20

[Shardanand 95] Upendra Shardanand & Pattie Maes. *Social information filtering: algorithms for automating" word of mouth"*. In Chi, volume 95, pages 210–217. Citeseer, 1995. 46

[Sharma 13] Lalita Sharma & Anju Gera. *A survey of recommendation system: Research challenges*. International Journal of Engineering Trends and Technology (IJETT), vol. 4, no. 5, pages 1989–1992, 2013. 25

[Sherkat 15] Ehsan Sherkat, Maseud Rahgozar & Masoud Asadpour. *Structural link prediction based on ant colony approach in social networks*. Physica A: Statistical Mechanics and its Applications, vol. 419, pages 80–94, February 2015. 28

[Shinde 12] Subhash K. Shinde & Uday Kulkarni. *Hybrid personalized recommender system using centering-bunching based clustering algorithm*. Expert Systems with Applications, vol. 39, no. 1, pages 1381–1387, January 2012. 44

[Singh 18] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik & Prasenjit Choudhury. *A Comparative Study of Different Similarity Metrics in Highly Sparse Rating Dataset*. In Data Management, Analytics and Innovation, pages 45–60. Springer, sep 2018. 80

[Sobecki 10] Janusz Sobecki & Jakub M Tomczak. *Student courses recommendation using ant colony optimization*. In Asian Conference on Intelligent Information and Database Systems, pages 124–133. Springer, 2010. 29

[Son 17] Jieun Son & Seoung Bum Kim. *Content-based filtering for recommendation systems using multiattribute networks*. Expert Systems with Applications, vol. 89, pages 404–412, 2017. 16

[Steck 13] Harald Steck. *Evaluation of recommendations: rating-prediction and ranking*. In Proceedings of the 7th ACM conference on Recommender systems, pages 213–220. ACM, 2013. 20

[Strub 16] Florian Strub, Romaric Gaudel & Jérémie Mary. *Hybrid recommender system based on autoencoders*. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pages 11–16. ACM, 2016. 16

[Sun 05] Xiaohua Sun, Fansheng Kong & Song Ye. *A comparison of several algorithms for collaborative filtering in startup stage*. In Proceedings of IEEE Networking, Sensing and Control., pages 25 – 28, 04 2005. 83

[Symeonidis 16] Panagiotis Symeonidis. *Matrix and tensor decomposition in recommender systems*. In Proceedings of the 10th ACM Conference on Recommender Systems, pages 429–430. ACM, 2016. 51

[Vargas 14] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou & Pablo Castells. *Coverage, redundancy and size-awareness in*

genre diversity for recommender systems. In Proceedings of the 8th ACM Conference on Recommender systems, pages 209–216, 2014. 53

[VOZALIS 07] M VOZALIS & K MARGARITIS. *Using SVD and demographic data for the enhancement of generalized Collaborative Filtering*. Information Sciences, vol. 177, no. 15, pages 3017–3037, August 2007. 19

[Wang 06] Jun Wang, Arjen P. de Vries & Marcel J. Reinders. *Unifying user-based and item-based collaborative filtering approaches by similarity fusion*. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 501–508. ACM, ACM Press, 2006. 39

[Wang 12] Yuanyuan Wang, Stephen Chi-Fai Chan & Grace Ngai. *Applicability of demographic recommender system to tourist attractions: a case study on trip advisor*. In Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03, pages 97–101. IEEE Computer Society, 2012. 15

[Wang 14a] Shanfeng Wang, Maoguo Gong, Lijia Ma, Qing Cai & Licheng Jiao. *Decomposition based multiobjective evolutionary algorithm for collaborative filtering recommender systems*. In Proceedings of IEEE Congress on Evolutionary Computation (CEC), pages 672–679. IEEE, jul 2014. 44

[Wang 14b] Zan Wang, Xue Yu, Nan Feng & Zhenhua Wang. *An improved collaborative movie recommendation system using computational intelligence*. Journal of Visual Languages & Computing, vol. 25, no. 6, pages 667–675, dec 2014. 15, 56, 85

[Wang 14c] Zan Wang, Xue Yu, Nan Feng & Zhenhua Wang. *An improved collaborative movie recommendation system using computational intelligence*. Journal of Visual Languages & Computing, vol. 25, no. 6, pages 667–675, dec 2014. 77, 86

[Wang 16a] Shanfeng Wang, Maoguo Gong, Haoliang Li & Junwei Yang. *Multi-objective optimization for long tail recommendation*. Knowledge-Based Systems, vol. 104, pages 145–155, 2016. 51

[Wang 16b] Zhongya Wang, Ying Liu & Steve Chiu. *An efficient parallel collaborative filtering algorithm on multi-GPU platform*. The

Journal of Supercomputing, vol. 72, no. 6, pages 2080–2094, 2016. 51

[Wasilewski 16] Jacek Wasilewski & Neil Hurley. *Incorporating diversity in a learning to rank recommender system*. In Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2016, Key Largo, Florida, USA, May 16-18, 2016., pages 572–578, 2016. 55

[Wu 16] Le Wu, Qi Liu, Enhong Chen, Nicholas Jing Yuan, Guangming Guo & Xing Xie. *Relevance Meets Coverage:A Unified Framework to Generate Diversified Recommendations*. ACM Transactions on Intelligent Systems and Technology, vol. 7, no. 3, pages 1–30, feb 2016. 54, 55

[Xie 15] Shengli Xie & Yifan Feng. *A Recommendation System Combining LDA and Collaborative Filtering Method for Scenic Spot*. In 2015 2nd International Conference on Information Science and Control Engineering. IEEE, apr 2015. 73

[Xue 05] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu & Zheng Chen. *Scalable collaborative filtering using cluster-based smoothing*. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 114–121. ACM, 2005. 15

[Yadav 18a] Sambhav Yadav, Vikesh Kumar, Shreyam Sinha & Sushama Nagpal. *Trust aware recommender system using swarm intelligence*. Journal of computational science, vol. 28, pages 180–192, 2018. 30

[Yadav 18b] Sambhav Yadav, Sushama Nagpal*et al. An Improved Collaborative Filtering Based Recommender System using Bat Algorithm*. Procedia computer science, vol. 132, pages 1795–1803, 2018. 29, 30

[Yan 19] Hangyu Yan & Yan Tang. *Collaborative Filtering Based on Gaussian Mixture Model and Improved Jaccard Similarity*. IEEE Access, vol. 7, pages 118690–118701, 2019. 85

[Yang 14] Xiwang Yang, Yang Guo, Yong Liu & Harald Steck. *A survey of collaborative filtering based social recommender systems*. Computer Communications, vol. 41, pages 1–10, March 2014. 17

[Yang 17] Bo Yang, Yu Lei, Jiming Liu & Wenjie Li. *Social Collaborative Filtering by Trust*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 8, pages 1633–1647, August 2017. 38

[Yin 12] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao & Chen Chen. *Challenging the long tail recommendation*. Proceedings of the VLDB Endowment, vol. 5, no. 9, pages 896–907, 2012. 11

[Yu 12] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si & Inderjit Dhillon. *Scalable coordinate descent approaches to parallel matrix factorization for recommender systems*. In 2012 IEEE 12th International Conference on Data Mining, pages 765–774. IEEE, 2012. 26

[Zahra 15] Sobia Zahra, Mustansar Ali Ghazanfar, Asra Khalid, Muhammad Awais Azam, Usman Naeem & Adam Prugel-Bennett. *Novel centroid selection approaches for KMeans-clustering based recommender systems*. Information Sciences, vol. 320, pages 156 – 189, 2015. 84, 107

[Zarzour 18] Hafed Zarzour, Ziad Al-Sharif, Mahmoud Al-Ayyoub & Yaser Jararweh. *A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques*. In 2018 9th International Conference on Information and Communication Systems (ICICS), pages 102–106. IEEE, 2018. 51

[Zhang 02] Tong Zhang & Vijay S Iyengar. *Recommender systems using linear classifiers*. Journal of machine learning research, vol. 2, no. Feb, pages 313–334, 2002. 14

[Zhang 08] Mi Zhang & Neil Hurley. *Avoiding monotony: improving the diversity of recommendation lists*. In Proceedings of the 2008 ACM conference on Recommender systems, pages 123–130. ACM, 2008. 52

[Zhang 09] Mi Zhang & Neil Hurley. *Novel item recommendation by user profile partitioning*. In 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, volume 1, pages 508–515. IEEE, IEEE, 2009. 56

[Zhang 16] Haijun Zhang, Tommy W. S. Chow & Q. M. Jonathan Wu. *Organizing Books and Authors by Multilayer SOM*. IEEE Transactions on Neural Networks & Learning Systems, vol. 27, no. 12, pages 2537–2550, December 2016. 85

[Zhang 18] Hao Zhang, Qiong Hong, Xiaomeng Shi & Jie He. *A Social Tagging Recommendation Model Based on Improved Artificial Fish Swarm Algorithm and Tensor Decomposition*. In Security with Intelligent Computing and Big-data Services, pages 3–13. Springer International Publishing, 2018. 30

[Zhang 19] Shuai Zhang, Lina Yao, Aixin Sun & Yi Tay. *Deep learning based recommender system: A survey and new perspectives*. ACM Computing Surveys (CSUR), vol. 52, no. 1, page p5, 2019. 35, 36

[Zhao 10] Zhi-Dan Zhao & Ming-Sheng Shang. *User-based collaborative-filtering recommendation algorithms on hadoop*. In Third International Conference on Knowledge Discovery and Data Mining, pages 478–481. IEEE, 2010. 26

[Zhao 14] Xin Wayne Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu & Xiaoming Li. *We know what you want to buy: a demographic-based system for product recommendation on microblogs*. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1935–1944. ACM, 2014. 15

[Zhou 10] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling & Yi-Cheng Zhang. *Solving the apparent diversity-accuracy dilemma of recommender systems*. Proceedings of the National Academy of Sciences, vol. 107, no. 10, pages 4511–4515, 2010. 52

[Zuo 15] Yi Zuo, Maoguo Gong, Jiulin Zeng, Lijia Ma & Licheng Jiao. *Personalized Recommendation Based on Evolutionary Multi-Objective Optimization [Research Frontier]*. IEEE Computational Intelligence Magazine, vol. 10, no. 1, pages 52–62, feb 2015. 56