Tampere University

STRATIGI MARIA

Group Recommendations with Responsibility Constraints

Tampere University Dissertations 848

Tampere University Dissertations 848

STRATIGI MARIA

Group Recommendations with Responsibility Constraints

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the auditorium K104
of the Linna building, Kalevantie 5, 33100, Tampere,
on 11 September 2023, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos	Associate Professor Konstantinos Stefanidis Tampere University Finland	
Supervisor	Professor Jyrki Nummenmaa Tampere University Finland	
Pre-examiners	Associate Professor Elisa Quintarelli University of Verona Italy	Associate Professor Mohamed Sharaf United Arab Emirates University United Arab Emirates
Opponent	Professor Letizia Tanca Politecnico di Milano Italy	

Cover design: Roihu Inc.

ISBN 978-952-03-3023-1 (print)
ISBN 978-952-03-3024-8 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-3024-8

HIILINEUTRAALI PAINOTUOTE
ClimateCalc CC-000025/FI
PunaMusta Printing

Carbon dioxide emissions from printing Tampere University dissertations have been compensated.

PunaMusta Oy – Yliopistopaino
Joensuu 2023

For my mum, σε ευχαριστώ για όλα

I love you

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to my two supervisors, Professors Konstantinos Stefanidis and Jyrki Nummenmaa, for their invaluable support and guidance during my doctoral studies. A special thank you to Professor Stefanidis for his encouragement over the past several years and the generous assistance he provided.

Also, I would like to express my gratitude to Evaggelia Pitoura, Haridimos Kondylakis, Katerina Tzompanaki, and Xiaozhou Li who were co-authors in the publications that led to the completion of this dissertation.

Finally, I would like to thank my family that always encourages me to go after the things I want and to never falter in the face of difficulties. Their steadfast support and love have made it possible for me to make my dreams a reality.

ABSTRACT

The expansion of social media has led more people to form groups for specific activities, and, consecutively, group recommender systems have emerged as popular research. In contrast to single recommendations, group recommendations involve a much greater degree of complexity since the systems are responsible for balancing the often conflicting interests of all group members. Due to the impact of recommendations on users' perceived performance (e.g., movie recommendations) and the often inherently sensitive nature of recommendation tasks (e.g., e-health recommendations), the process by which recommendations are generated should be carefully considered. As a result, it has become increasingly necessary to develop recommendations that adhere to various responsibility constraints. Such responsibility constraints include *fairness*, which corresponds to a lack of bias, and *transparency*, which facilitates an understanding of the processes of the system.

Nevertheless, if these constraints are followed, group recommender systems become more complex. It is even more challenging if they are to consider a sequence of recommendations rather than each recommendation as a separate process. Intuitively, the system should take into account the historical interactions between itself and the group and adjust its recommendations in accordance with the impact of its previous suggestions. This observation leads to the emergence of a new type of recommender system, called *sequential group recommendation* systems. However, standard group recommendation approaches are ineffective when applied in a sequential scenario. They often produce recommendations that are not even intended to be fair to all group members, i.e., not all group members are equally satisfied with the recommendations. In practice, when each recommendation process is considered in isolation, there is always going to be a least satisfied member. However, the least satisfied member should not always be the same when the scope of the system encompasses more than one recommendation round. This will result in the fairness constraint being broken since the system is biased against one group member.

As a result of the complex nature of recommender systems, users may be unable to understand the reasoning behind a suggestion. To counter this, many systems provide explanations along with their recommendations in adherence to the transparency constraint. Discussing why not suggesting an item is valuable, especially for system administrators. Explanations to such queries are invaluable feedback for them when they are in the process of calibrating or debugging their system.

Overall, this thesis aims to answer the following Research Questions (RQ). *RQ1. How to define sequential group recommendations, and why are they needed? How to design group recommendation methods based on them?* This thesis formally defines a sequential group recommender system and what objectives it should observe. Additionally, it proposes three novel group recommendation methods to produce fair sequential group recommendations. *RQ2. How to exploit reinforcement learning to select a group recommendation method when the system's environment changes after each recommendation round?* In an extension of the RQ1, this thesis proposes a reinforcement-based model that selects the most appropriate group recommendation method to apply throughout a series of recommendations while aiming for fair recommendations. *RQ3. How to design questions and produce explanations for why a set of items did not appear in a recommendation list or at a particular position?* This dissertation defines what a Why-not question is, as well as presents a structure for them. Additionally, it proposes a model to generate explanations for these Why-not questions. *RQ4. How to incorporate various health-related aspects in group recommendations?* It is important to make fair recommendations when dealing with extremely sensitive health-related information. In order to produce as fair a recommendation as possible, this thesis proposes a model that incorporates various health aspects.

TIIVISTELMÄ

Sosiaalisen median laajeneminen on johtanut siihen, että yhä useammin ihmiset muodostavat ryhmiä erilaisia aktiviteetteja varten, ja peräkkäisiä ryhmäsuositteluja tuottavat järjestelmät ovat nousseet suosituksi tutkimusalueeksi. Ryhmälle tehtävät suositukset ovat huomattavasti monimutkaisempia kuin yksittäiset suositukset, koska suosittelujärjestelmät joutuvat vastaamaan kaikkien ryhmän jäsenten usein ristiriitaisten etujen tasapainottamisesta. Ottaen huomioon suositusten vaikutus käyttäjien kokemaan järjestelmän suorituskykyyn (esim. elokuvasuositukset) ja suositustehtävien usein varsin arkaluontoinen luonne (esim. sähköisen terveydenhuollon suositukset), suositusten luomisprosessia tulee harkita huolellisesti. Näistä seikoista johtuen on tullut entistä tarpeellisemmaksi kehittää erilaisia vastuullisuusrajoitteita noudattavia suosituksia. Tällaisia vastuullisuusrajoitteita ovat muun muassa *reiluus* eli puolueettomuus, ja *läpinäkyvyys*, joka helpottaa järjestelmän prosessien ymmärtämistä.

Jos näitä rajoituksia noudatetaan, niin ryhmäsuosittelijoista tulee monimutkaisempia. On edelleen haastavampaa, jos suosittelijat käsittelevät suositusten jonoa sen sijaan, että jokainen suositus käsitellään erillään muista. Intuitiivisesti järjestelmän tulee ottaa huomioon itsensä ja ryhmän välisen vuorovaikutuksen historia ja mukauttaa suosituksiaan aikaisempien suositusten vaikutuksen mukaisesti. Tämä havainto johtaa uuden suositusjärjestelmätyypin, *peräkkäisten ryhmäsuositusjärjestelmien*, syntymiseen. Tavalliset ryhmäsuositusmenetelmät ovat tehottomia, kun niitä käytetään peräkkäisessä skenaariossa. Ne tuottavat usein suosituksia, joita ei ole edes tarkoitettu reiluksi kaikkia ryhmän jäseniä kohtaan, eli kaikki ryhmän jäsenet eivät ole yhtä tyytyväisiä suosituksiin. Käytännössä, kun jokaista suositusprosessia tarkastellaan erikseen, aina löytyy vähiten tyytyväinen jäsen. Vähiten tyytyväisimmän jäsenen ei kuitenkaan pitäisi aina olla sama, kun järjestelmän käyttö kattaa useamman kuin yhden suosituskierroksen. Tämä johtaisi oikeudenmukaisuuden rajoitteen rikkomiseen, koska järjestelmä olisi puolueellinen yhtä ryhmän jäsentä vastaan.

Suositusjärjestelmien monimutkaisuuden vuoksi käyttäjät eivät ehkä pysty ym-

märtämään ehdotuksen perusteluja. Tämän torjumiseksi monet järjestelmät tarjoavat selityksiä ja suosituksia avoimuusrajoituksen mukaisesti. Keskustelu siitä, miksi kohdetta ei ehdoteta, on arvokasta erityisesti järjestelmänvalvojille. Selitykset tällaisiin kyselyihin ovat heille korvaamatonta palautetta, kun he ovat kalibroimassa tai korjaamassa järjestelmäänsä.

Kaiken kaikkiaan tämän opinnäytetyön tavoitteena on vastata seuraaviin tutkimuskysymyksiin (RQ). *RQ1. Kuinka määritellä peräkkäiset ryhmäsuositukset ja miksi niitä tarvitaan? Kuinka suunnitella ryhmäsuositusmenetelmiä niiden pohjalta?* Tässä opinnäytetyössä määritellään formaalisti peräkkäinen ryhmäsuositusjärjestelmä ja mitä tavoitteita sen tulee noudattaa. Lisäksi ehdotetaan kolmea uutta ryhmäsuositusmenetelmää oikeudenmukaisten peräkkäisten ryhmäsuositusten tuottamiseksi.

RQ2. Kuinka hyödyntää vahvistusoppimista ryhmäsuositusmenetelmän valinnassa, kun järjestelmän ympäristö muuttuu jokaisen suosituskierroksen jälkeen? RQ1:n laajennuksessa tässä opinnäytetyössä ehdotetaan vahvistukseen perustuvaa mallia, joka valitsee sopivimman ryhmäsuositusmenetelmän käytettäväksi koko sarjassa, samalla pyrkien reiluuteen.

RQ3. Kuinka suunnitella kysymyksiä ja tuottaa selityksiä sille, miksi jokin joukko ei näkynyt suosituslistalla tai tietyssä paikassa? Tässä väitöskirjassa määritellään miksi-ei-kysymys ja esitetään näiden kysymysten rakenne. Lisäksi työssä ehdotetaan mallia, jolla luodaan selityksiä näihin miksi-ei-kysymyksiin.

RQ4. Kuinka sisällyttää erilaisia terveyteen liittyviä näkökohtia ryhmäsuosituksiin? Näissä on tärkeää antaa oikeudenmukaisia suosituksia, koska terveyssuositukset ovat erittäin arkaluontoisia. Mahdollisimman oikeudenmukaisen suosituksen tuottamiseksi tässä opinnäytetyössä ehdotetaan mallia, joka sisältää erilaisia terveysnäkökohtia.

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

BGEM	Bipartite Graph Embedding Model
CF	Collaborative Filtering
DFH	Discounted First Hit
e.g.	for example
et al.	and others
etc.	et cetera
glossary	list of terms and symbols
GRS	Group Recommender Systems
i.e.	that is
ICD10	International Statistical Classification of Diseases and Related Health Problems
LCA	Lowest Common Ancestor
MDP	Markov Decision Problem
ML	Machine Learning
NDCG	Normalized Discounted Cumulative Gain
PHR	Personal Health Records
RL	Reinforcement Learning
RQ	Research Question
RS	Recommender Systems
SDAA	Sequential Dynamic Adaptation Aggregation
SIAA	Sequential Individual Adaptation Aggregation

SQUIRREL	SeQUentIal Recommendations with ReinforcEment Learning
VAE	Variational Autoencoders

ORIGINAL PUBLICATIONS

Publication I — Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. "Fair sequential group recommendations". In: *SAC '20: The 35th ACM/SIGAPP Symposium on Applied Computing, online event, [Brno, Czech Republic], March 30 - April 3, 2020*. Ed. by Chih-Cheng Hung, Tomás Cerný, Dongwan Shin, and Alessio Bechini. ACM, 2020, pp. 1443–1452. DOI: 10.1145/3341105.3375766. URL: https://doi.org/10.1145/3341105.3375766.

Publication II — Maria Stratigi, Evaggelia Pitoura, Jyrki Nummenmaa, and Kostas Stefanidis. "Sequential group recommendations based on satisfaction and disagreement scores". In: *J. Intell. Inf. Syst.* 58.2 (2022), pp. 227–254. DOI: 10.1007/s10844-021-00652-x. URL: https://doi.org/10.1007/s10844-021-00652-x.

Publication III — Maria Stratigi, Evaggelia Pitoura, and Kostas Stefanidis. "SQUIRREL: A framework for sequential group recommendations through reinforcement learning". In: *Inf. Syst.* 112 (2023), p. 102128. DOI: 10.1016/j.is.2022.102128. URL: https://doi.org/10.1016/j.is.2022.102128.

Publication IV — Maria Stratigi, Katerina Tzompanaki, and Kostas Stefanidis. "Why-Not Questions & Explanations for Collaborative Filtering". In: *Web Information Systems Engineering - WISE 2020 - 21st International Conference, Amsterdam, The Netherlands, October 20-24, 2020, Proceedings, Part II*. Ed. by Zhisheng Huang, Wouter Beek, Hua Wang, Rui Zhou, and Yanchun Zhang. Vol. 12343. Lecture Notes in Computer Science. Springer, 2020, pp. 301–315. DOI: 10.1007/978-3-030-62008-0_21. URL: https://doi.org/10.1007/978-3-030-62008-0%5C_21.

Publication V Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. "Multidimensional Group Recommendations in the Health Domain". In: *Algorithms* 13.3 (2020), p. 54. DOI: 10.3390/a13030054. URL: https://doi.org/10.3390/a13030054.

Author's contribution

The contributions of Maria Stratigi to each of the above-mentioned publications are specified as follows.

Publication I The author played a significant role in the study's planning and execution, including conception, data collection, formal analysis, methodology, and evaluation under the supervision of the co-authors. The author also was in charge of the writing. Co-authors contributed to the editing and review process. The author is the publication's lead author.

Publication II The author played a significant role in the study's planning and execution, including conception, data collection, formal analysis, methodology, and evaluation under the supervision of the co-authors. In addition to writing the publication, the author had the assistance of the co-authors in editing and reviewing it. The author is the publication's lead author.

Publication III The author was in charge of planning and implementing the study, researching the related works, designing the SQUIRREL framework solution, and performing the experimental evaluation. The author was also leading the writing process. The author is the publication's lead author.

Publication IV The author was in charge of the conceptualization, data curation, formal analysis, and methodology of the proposed solution. The author also implemented and evaluated the proposed solution and took control of writing most of the manuscript under the guidance of the co-authors. The author is the publication's lead author.

Publication V	Author was responsible for planning the study in terms of conceptualization, data curation, formal analysis, methodology, implementation, and validation under the supervision of the co-authors. In addition to writing, the author worked with the co-authors on editing and reviewing the manuscript. The author is the publication's lead author.

1 INTRODUCTION

In recent years, Recommender Systems (RS) have been one of the most active research areas in computer science. They play a key role in the majority of applications aimed at end-users. RS are used to enhance user experiences across a range of activities, from listening to music and movie recommendations to finding health information or hiring an employee. At the same time, it is also simpler to organize groups to carry out an activity, mainly due to the growth of social media. They encourage people from many social groups, from close friends or families to strangers, to form a group and engage in an activity. Consequently, Group Recommender Systems (GRS) are a significant area of study [4, 59, 60]. A standard example of a GRS is a group of friends that want to watch a movie. Every friend has preferences that the other friends might not share. The goal of a GRS is to balance the preferences of all group members and generate a list of items to recommend to the group. There are various methodologies for producing such a list [37]. One of the most popular is applying a single RS for all group members and then aggregating their corresponding recommendation lists into one list for the group (e.g., [4, 5]).

Furthermore, in various cases, irresponsible recommendation techniques have been cited as having counter-effects and being untrustworthy. Unfair treatment of different users, non-transparency and extensive personalization based on users' data can reduce users' trust in the system. To encourage users to trust an RS, it must adhere to several responsibility constraints, such as fairness, non-discrimination, or transparency [83]. There is a wide range of definitions for each of these constraints, and they can be implemented in different ways. Generally, fairness corresponds to a lack of bias towards users and data items; the non-discrimination constraint encourages diversity in the results, and transparency enables users to better understand the system's inner workings.

This dissertation mainly focuses on two responsibility constraints; *fairness* and *transparency*. The notion of fairness in group recommendations can be vague with multiple interpretations [64]. Nonetheless, a general description of a *fair* group recommender system is a system that lacks bias against any group member. Simple solutions will not work for this issue, which demands a complicated solution. For example, a fundamental approach to achieve fairness is to utilize the *Average* aggregation method, in which the score of an item in the group recommendation list equals its average score from the group members' individual recommendation lists. All group members are viewed as equals in this regard. This approach, nevertheless, has a severe flaw that is better demonstrated with the following example. Consider a group with three members where the first two share mutual tastes, but the third does not. The *Average* approach has a tendency to disregard the third user's feedback. As such, the group recommender is biased against the third member.

The problem of fair group recommendations is complex enough, but it becomes even more complicated to achieve when the system needs to consider multiple recommendation rounds instead of just one. In most cases, a GRS perceives each interaction with the group as a standalone process. However, a group normally uses a GRS multiple times. The group's reaction to the system's recommendations in previous recommendation rounds is an important indicator that would be advantageous for the system to consider. Thus, a group recommender system should take into account a *sequence* of interactions with the group. A *sequential group recommender* should record all recommendations provided to the group, along with group members' satisfaction with them, and consider both when producing suggestions for the next round. Multiple recommendation rounds add a new layer of complexity for achieving fair group recommendations. Supposing each group member experiences a degree of satisfaction with a provided group recommendation list, the system should provide recommendations without bias against any group member throughout a series of recommendation rounds. In other words, the same group member should not always be the least satisfied. The primary focus of this thesis is on stable groups, i.e., groups whose members remain constant over time. However, it also examines the effect of ephemeral groups on sequential recommender systems. Ephemeral groups experience change after each recommendation round, either with the addition of new members or the departure of established members.

In this dissertation, we also examine *transparency* in recommendations, commonly referred to as explanations in recommendations. Although recommenders make an effort to make relevant suggestions to users based on their preferences, they often cannot find the most relevant data items to offer. This causes the users to lose faith in the system, which frequently results in them refusing to use it or falsely believing there is a problem with it. Providing explanations in recommender systems is a recurring problem in the research community. The most common solution is to provide explanations in conjunction with recommendations so that the user or the person who designed the system can then gain an understanding of the rationale behind a suggestion [103, 82, 18]. The level of detail and the presentation style of the explanations can differ depending on the end-user, i.e., the final user or the system administrator.

Users can be presented with explanations of recommendations automatically or after providing feedback [28]. In the latter case, the feedback often takes the form of a query from the user to the system (e.g., [71, 34]). Such questions can be either positive or negative concerning the existence of a data item in the recommendation list. For example, a question can be either why an item is suggested or why an item was not suggested. Many works explore the first case. However, questions concerning the non-existence of a data item remain unexplored. The implicit constraint of such Why-not questions is that the user knows the system's database and, thus, questions why an expected item was not suggested. To this end, Why-not questions and their explanations are mainly geared toward system administrators. It is necessary to ask these kinds of questions because they give a system engineer a deeper understanding of the system and how to debug it.

1.1 Research Goals and Questions

An RS can be applied to various problems, from innocent daily dilemmas, such as choosing shoes, to more significant choices, such as who to hire. Regardless of the importance of the suggestion provided by an RS, the system should behave responsibly. In our work, we focus on ensuring fairness and provide explanations along with recommendations. Although both have been extensively researched, some gaps still exist. The goal of this dissertation is to resolve the following research gaps.

Sequential group recommendations are a relatively new area of research. This dissertation formally defines the sequential group recommendation problem and explores how to achieve fair sequential group recommendations. By the nature of sequential recommendations, the time dimension is considered in the recommendation process. This means that the data available to the system change between rounds of recommendation. After each round, additional information (i.e., ratings, reviews, etc.) is included in the system. More significantly, the information about the group changes. For instance, the group members' satisfaction with the system changes following a round of recommendations. A system should adapt to these changes and still produce fair recommendations. This thesis proposes a recommendation model based on reinforcement learning to overcome this challenge.

Additionally, when developing a recommender system for a sensitive domain such as health, the system should produce suggestions that are as relevant as possible. To this end, the system should take into consideration multiple sources of information, such as the users' health problems, their health literacy, and their interests in medical documents. The thesis proposes a group recommendation model specifically designed for the health domain. Finally, the ability to automatically explain why a data item was not displayed as expected is an important tool for the developer when calibrating their system. This dissertation proposes a formalization of Why-not questions and offers a method to provide their corresponding explanations.

Therefore, this thesis, considering all the research gaps mentioned, shall contribute to answering the following research questions (RQ).

RQ1. How to define sequential group recommendations and why are they needed? How to design group recommendation methods based on them?

Sequential group recommendations are a significant research area since the formation of groups is especially prevalent in recent times. This is mainly due to the expansion of social media, which enables users to form groups more easily and efficiently. These groups interact with a recommender system multiple times, so the system should consider the previous interactions and adjust its behavior according to the effectiveness of the previous recommendation rounds.

In the scope of this work, a sequential group recommender system is a system that offers recommendations to a group for a series of recommendation rounds. To produce a recommendation for the group at each round, all previous interactions between the system and the group should be considered. Generally, a sequential group recommender system should offer good results for the current round of recommendations and simultaneously satisfy the group members over a sequence of recommendations. More specifically, the system should observe two objectives. First, the recommended items should be as relevant as possible to the group members at each round. Second, the system needs to ensure that no group member is dissatisfied after a sequence of recommendations. Meaning that no group member is continuously offered items that are of no interest to them. Standard group recommendation approaches are not enough to achieve both sequential group recommendation objectives. However, specifically designed sequential group recommendation approaches, which consider the previous rounds of recommendation, show a significant increase in their performance compared to the standard group recommendation methods. This work showcases three such methods and explores their effectiveness in various test cases.

RQ2. How to exploit reinforcement learning to select a group recommendation method when the system's environment changes after each recommendation round?

The information available to the system rapidly changes over time, with additional data being augmented. As a result, the interests of the group members may change with the additional feedback. Based on the severity of these changes, the effectiveness of a recommendation model may be affected. Ideally, the recommendation model should observe these changes and modify its behavior accordingly. This work proposes a model based on reinforcement learning techniques since it mirrors the sequential group recommendation problem. As a reinforcement learning model, it consists of three parts, environment, actions, and reward. The criteria for designing each model part are flexible based on what the model wants to achieve. This work utilizes the following: at each recommendation round, the system considers the environment, i.e., the satisfaction of the group members, and selects an appropriate action. The actions component of the reinforcement learning model lends itself naturally to the various available aggregation methods. The application of an action has two effects. First, each action changes the environment. Second, a reward is calculated. The reward function is flexible and can focus on what the system wants to achieve. For example, if the system wants to maximize the users' satisfaction, then a reward function can be the overall group satisfaction.

RQ3. How to design questions and produce explanations for why a set of items did not appear in a recommendation list or at a particular position?

RS are often a black box for the end-users. Explanations are useful when these users do not receive an item at the expected ranking in the list or do not receive the item at all. Explanations on why an item did not appear in the recommendation list ensure the users that there are no errors with the system and describe the reasoning behind the produced results. These explanations are incredibly informative for system administrators when calibrating or debugging the system. The system should answer these questions without human intervention. To this end, since these questions have to be processed by a system, they have to have a formalized structure. This work focuses on providing a system administrator with a user-friendly explanation of their system; hence, the explanations are based on the model and its different variables.

RQ4. How to incorporate various health-related aspects in group recommendations?

RS in the health domain are a powerful tool for patients that search for medical documents on the web about their health problems. Moreover, it has been demonstrated that utilizing group dynamics is highly effective, for example in lowering relapse rates in smoking. In therapy sessions, caregivers provide their patients with access to more appropriate online resources to manage their health problems. They encourage users to take a more proactive approach to their health. Due to the sensitive nature of the health domain, the system should provide users with documents as relevant as possible to their interests. Therefore, the system must consider information about the users' health problems, health literacy, and other factors. It is possible to include this information in various aspects of the recommendation model. This work is focused on a group recommendation model and incorporates the users' health information in a novel similarity function.

1.2 Contributions

The scope of this work contributes to multiple areas of recommender systems. The dissertation consists of five published peer-reviewed articles (Publication I-V). Publications I and II address the RQ1 by defining the sequential group recommendation problem and proposing three novel group recommendation methods designed explicitly for sequential group recommendations. Publication III answers RQ2 by presenting a group recommendation model based on reinforcement learning. RQ3 is expanded on Publication IV, where the Why-not questions and explanations for recommendations are demonstrated. Publication V answers the RQ4, wherein a novel group recommender system encompasses the multidimensional aspects of the health domain. The contributions of each article are summarized as follows.

Publication I: Fair sequential group recommendations.

Publication I [86] first introduces the sequential group recommendation problem and the notion of satisfaction and disagreement. It proposes a new group recommendation method that aims to provide a recommendation list that is both satisfying and fair to all group members. This method was extensively evaluated in a series of experiments using the MovieLens 20M dataset [32].

Publication II: Sequential Group Recommendations based on Satisfaction and Disagreement Scores.

Publication II [88] is an expansion on the previous, Publication I. It further refines the sequential group recommendation problem and proposes two additional group recommendation methods. The evaluation is more extensive by utilizing an additional real-world dataset, GoodReads [96]. In addition, it extends the sequential group recommendation scenario by including ephemeral groups, i.e., groups that change their members between rounds of recommendations. It evaluates the effect such groups have on the proposed sequential group recommendation methods.

Publication III: SQUIRREL: A Framework for Sequential Group Recommendations through Reinforcement Learning.

Publication III [89] proposes a sequential group recommendation model based on reinforcement learning consisting of three main elements, environment, actions, and reward. The environment depicts how satisfied each group member is. The actions are various group recommendation methods, i.e., the three proposed in Publications I and II and another three state-of-the-art methods. In addition, the publication examines two different reward functions. The experimental procedure is further expanded by adding a third real-world dataset, Amazon [33].

Publication IV: Why-Not Questions & Explanations for Collaborative Filtering.

Publication IV [90] contribution is formulating questions to a system about why an item does not appear in an expected position in the recommendation list and providing the corresponding explanation. To create such Why-not questions that can be used in a wide range of scenarios, this study proposes a structure and properties that they should possess in order to be able to handle complex questions. Additionally, a method is proposed to generate explanations of such questions based on the initial recommendation model utilized to generate the recommendation list. A series of evaluations show the effectiveness of the method mentioned above.

Publication V: Multidimensional Group Recommendations in the Health Domain.

In Publication V [85], the contribution is designing a GRS that incorporates complex and multidimensional aspects of the health domain. The sensitive nature of such recommenders makes the construction of each component more challenging. The system needs to provide relevant items to the end-users while adhering to their health problems, literacy levels, and psycho-emotional status. To address this issue, this publication proposes a similarity function and a new method for fair group recommendations. Additionally, the creation of two synthetic datasets is described in this work. Extensive evaluation using these datasets shows that the group recommendation model enhanced with various health-related aspects is more effective than a standard approach.

1.3 Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 provides an overview of the relevant research areas, including group and sequential recommendations, reinforcement learning in recommender systems, explanations, and recommendations in the health domain. Chapter 3 describes the sequential group recommendation problem and the methods proposed for it, while Chapter 4 introduces the SQUIRREL model designed for achieving sequential group recommendations through reinforcement learning techniques. Chapter 5 proposes the formalization of Why-not questions for recommender systems and the methods to provide explanations for them, and Chapter 6 describes a group recommendation model that incorporates multiple aspects of the health domain. Finally, Chapter 7 concludes the thesis and discusses future work directions. At the end of the thesis, the original research publications are attached in their original format.

2 RELATED WORK

2.1 Recommender Systems Overview

Using already available user information a Recommender System (RS) will suggest to a user a list of items that are of potential interest to them. The general requirement for an RS is a collection of items I and a user set U. Every user provides feedback, for a subset of items I. In many cases, this feedback takes the form of ratings, reviews, clicks, etc. The set of users that provided feedback for an item i is defined by U_i, while the list of items that a user u has given feedback on is denoted by I_u. RS estimates a prediction score $p(u, i)$ for item i that user u has not given any feedback on. The user's recommendation list will be composed of the items with the highest prediction scores.

Several studies have been conducted on estimating an item's prediction score for a user [1]. The various works that comprise this thesis are based on *Collaborative Filtering* (CF), which is a well-established approach to recommending items (e.g., [47, 15, 60]) and for simplicity unless otherwise specified the user feedback is provided in the form of ratings. CF algorithms are generally categorized as memory-based or model-based [8]. The memory-based CF algorithms are further distinguished into user-based and item-based. User-based algorithms identify users who are similar to the target user by using a similarity function. The prediction is determined by analyzing similar users' ratings, or as they are commonly referred to as *neighbors or peers*. Item-based CF algorithms suggest to users items that are similar to items they have previously enjoyed. Model-based CF algorithms primarily use machine learning techniques to discover the implicit behavior of users [91]. Matrix factorization is one of the most commonly used model-based methods [52]. Its aim is to factorize a user-item matrix into two low-ranked matrices, the user-factor matrix and the item-factor matrix, that can identify potential items that users may find interesting. Several other model-based approaches have been proposed, including neural networks [72], which are more flexible than matrix factorization and can handle a more significant amount of data, fuzzy systems [100], which can address the underlying uncertainties, lack of precision, and ambiguity in features of items and user's behavior, and Bayesian classifiers [16], which transform the recommendation problem into a classification task.

Fairness in RS. Fairness in recommender systems has been the subject of multiple studies [65, 11, 10, 66, 26, 53]. To our knowledge, there has yet to be a formalized and widely accepted definition of fairness in recommendations. In contrast, many works offer their own definition of fairness. It is possible, however, to generalize and categorize fairness definitions. Fairness definitions can be distinguished into two main categories: individual and group fairness. In individual fairness, equal treatment is premised on similar entities being treated equally. Group fairness definitions use protected attributes to define groups; all groups should receive the same treatment.

The authors of [101] explore the issues surrounding fairness in CF recommender systems, which are sensitive to discrimination based on historical data. Since predictions are based on empirical data, they may inherit any existing bias. They consider two groups; advantaged and disadvantaged. For example, female presentation percentages are lower (disadvantaged) in STEM courses when compared to male presentations (advantaged). They propose various unfairness metrics based on the discrepancy between predicted and actual scores for items suggested to these two groups. The metric of *value unfairness* refers to the differences in estimation errors between groups, i.e. when one group receives a prediction that is greater or lower in value than their actual preference. The *absolute unfairness* metric calculates discrepancies in absolute estimation errors within groups. *Underestimation unfairness* measures the discrepancy between the predictions and the true ratings. Finally, *non-parity unfairness* determines the difference between average ratings for disadvantaged and advantaged users.

In [7], the proposed pairwise fairness method considers the items' relative positions in the recommendation list rather than their scores. Two groups of items are considered: protected and non-protected. *Pairwise accuracy* measures how likely it is that a clicked item will rank above an unclicked item for the same user. Subsequently, the *pairwise fairness* is achieved if the groups have the same *pairwise accuracy*. Furthermore, they distinguish between fairness within (intra-group) and between groups (inter-group). *Intra-group pairwise fairness* is achieved if an item is ranked higher than a relevant unclicked item from the same group, regardless of which group it belongs to. When comparing items across groups, *inter-group fairness* occurs if a clicked item is ranked higher than a relevant unclicked item that belongs to the other group.

[11] utilizes Variational Autoencoders (VAE) as a CF method and addresses the issue of providing fair recommendations when the data information is unavailable. A particular focus is placed on the influence of position bias, where items with remarkably similar scores are placed in different rankings in the recommendation list, and popularity bias, where unpopular items are less likely to be suggested. Popularity is also taken into account in [24], where they investigate the long-term fairness of recommendations when attributes, like popularity, are subject to change with time as a result of recommendation policies and user feedback. Instead of examining how to achieve fairness for existing items, [109] examines the fairness of new items when they are first introduced to the system, i.e., a cold-start scenario. The authors propose an innovative framework for learnable post-processing for achieving fairness for newly introduced items, in addition to two models derived from it.

Unlike previous works that examined fairness from the consumer's perspective, [81] proposes a calibration-based approach to fairness on the producer side. A calibrated recommendation algorithm will provide recommendations that reflect a user's interests and are proportional to them. Ideally, a recommendation list should have a similar proportion of items from different groups as those in the user's history. Let's say that the recommended items are movies and that the protected attribute is the genre. To calculate the fairness of the recommendations, they compare the distributions of a specific genre in the recommendations and the user's history. A re-ranking algorithm is introduced in [27] that emphasizes fairness for providers in accordance with a notion of equity that has been taken from the provider's viewpoint. The algorithm re-distributes the recommendation among the providers, with a share of the recommendation being allocated to each provider according to their representation in the input data.

2.2 Group Recommendations

The research behind group recommendations is extensive [54]. Generally, group recommendations can be divided into two main categories [37]. The first approach involves combining the group members' ratings to form a virtual user that can be used in standard recommendation algorithms (e.g., [55, 104]). The second and more commonly used method for group recommendations is to use the same single-user RS for each group member (e.g., [4, 5, 59]). Subsequently, the system aggregates each member's list into one to generate a group recommendation list. All of the works presented in this thesis follow the second approach, as it allows for increased flexibility and provides avenues for further development in terms of efficiency and effectiveness [61].

A group recommender system may take into account a wide variety of criteria at the aggregation stage. As an example, the work presented in [105] suggests a group recommendation model that considers the influence of each group member during the aggregation process. Members with more knowledge of the recommended items have the most influence, meaning their weight during aggregation is higher. Additionally, [17] demonstrates how the aggregation strategy can be deduced from recent advances in attention networks and collaborative filtering. Similarly, [102], besides using an attention mechanism, utilizes a Bipartite Graph Embedding Model (BGEM) to estimate each member's contribution to the group's eventual decision. [74] adopts a preference-based social network and analyzes social connections between group members to make a final decision without knowing each member's preferences.

In [95], the most effective aggregation strategy is determined through an analysis of the interaction between group members. Social self-attention networks were used to determine the voting patterns of the group members based on the multiple voting processes that simulate how consensus is reached. [67] provides a novel solution to the problem of producing recommendations for large groups by subdividing a large group of people according to their own interests. Specifically, it identifies media-user pairs that are likely to be candidates for each subgroup and combines their recommendations lists. [43] outlines a two-phase recommendation system that seeks to satisfy each member of the group. The first phase involves satisfying the whole group. The second phase involves filtering out irrelevant items for each member in order to satisfy them individually.

Fairness in Group Recommendations. Fairness in group recommendations is a widely researched area. However, it is essential to note that group recommendations suffer from the same pitfalls as single recommendations when it comes to defining fairness. There is no formal definition, and each work presents a new perspective on group fairness.

In [48], fairness is presented as a constrained optimization problem. In particular, for a given set of rankings (i.e., group members' individual recommendation lists), the method provides the most similar ranking to the provided sets that satisfy a specific fairness requirement. Assuming that all data items have a protected property that separates the dataset into *n* distinct groups, the authors propose as a fairness requirement a general formulation of statistical parity for rankings that takes into account the top-k prefix of the rankings, which is the top-k parity.

Among the various methods for assessing fairness, [99] measures the degree of satisfaction, or utility, for each group member with the group recommendation list, based on the relevance of the recommended items for each member, i.e., what is the predicted score of each recommended item in the user's personal recommendation list. Specifically, the utility of a set of items for a user can be expressed as the average utility of these items. Overall user satisfaction for a group recommendation list, referred to as social welfare, is determined by averaging all group members' utilities. Then, fairness is calculated by comparing the group members' utilities. Generally, the fairest list would minimize user dissatisfaction within the group. Therefore, fairness imposes the least misery policy with regard to users' utilities, placing emphasis on the disparity between the lowest and highest. In accordance with this idea, fairness can be defined as the minimum utility among group members or by the variance in them when encouraging group members to attain similar utility values.

In contrast to using group members' utility for group recommendation, [73] uses the position of the items in the group recommendation list. The authors proposed approach for fair group recommendations is based on the concept of Pareto optimality: an item *i* is Pareto optimal if none of the other items *j* rank higher than it, i.e., no item *j* dominates item *i*. Subsequently, N-level Pareto optimal is a variation of Pareto optimal consisting of items dominated by N-1 other items and identifying the N best items to be recommended. By definition, an item set of this type is fair since each user's top choices are included. Similar to this, to facilitate the aggregation phase, [41] introduces the concept of rank-sensitive balance. The group's first recommendation should strive to balance all members' interests to the maximum extent possible. In the same way, the first two items together must also accomplish the same task, etc.

Recent work presented in [77] has offered two definitions of fairness: *fairness proportionality* and *envy-freeness*. In *fairness proportionality*, when the user *u* likes at

Table 2.1 Fairness definitions taxonomy in group recommendations. The works in **bold** are works presented in this dissertation.

	Individual	Group	Consumer	Producer	Single Round	Multiple Rounds
[48]		✓	✓		✓	
[99]	✓		✓		✓	
[73]	✓		✓		✓	
[77]	✓		✓		✓	
[87]	✓		✓			✓
[88]	✓		✓			✓
[89]	✓		✓			✓
[85]	✓		✓		✓	

least m items in the recommended list, the user considers the list fair for them. In *envy-freeness*, if the recommendation list contains at least m items for which the user does not feel envious, then u will consider the list fair. Despite the fact that these resulting suggestions may not be the most optimal, they are nevertheless fair since there is at least one item on the list that appeals to each member of the group.

The voting theory was used as a basis for some of the first approaches to achieving fairness in group recommendations [65, 63]. For example, [57] uses voting theory to decide which item to recommend using probability knowledge about user ratings. An alternative approach is proposed by [29] in which members of the group are allowed to comment on each other's choices. It allows users to receive new recommendations that are similar to those made by other members of the group as well as to present their own counter-proposals that explain their reasoning.

The ranking aggregation methods proposed in these works can be adapted to work as additional actions in the reinforcement learning model proposed in this thesis, SQUIRREL [89]. It may be necessary to make additional modifications to the model in order to satisfy the requirements of the added method, like altering the state or reward definition or providing other input. Furthermore, depending on how complex the aggregation method is, it can be time-consuming to retrain the model; the basic Average method requires less training time than a sophisticated group recommendation method.

Overall, Table 2.1 categorizes the works based on the type of fairness they achieve on three different dimensions. Individual and group fairness, consumer or producer-side fairness, and if the works consider single or multiple rounds of recommendations. In individual fairness, similar entities are treated equally. In group fairness, all groups should receive the same treatment. It is worth mentioning that research literature has heavily focused on consumer-side fairness, and no works currently examine how to achieve fairness in group recommendations from a producer's perspective.

2.3 Sequential Recommendations.

Based on the number of past interactions they consider, sequential recommenders can be classified into three broad categories: *Last-N interactions-based recommendations*, *Session-based recommendations* and *Session-aware recommendations* [69]. In Last-N interactions-based recommendations, only the user's most recent N actions are taken into consideration [20, 49, 50]. Generally, this occurs as a result of the sheer volume of historical records stored by the system per user - those records are often duplicated and provide no useful information - which overwhelms the system. Session-based recommendations are only made based on interactions that have occurred during the current session. Most commonly, they appear in news articles and advertisements [23, 35]. Session-aware systems have information on the user's history and the last interaction with the system. There is a widespread use of these recommenders in e-commerce and app recommendations [31, 38, 70].

The development of a session-aware music recommender system is discussed in [30]. In a neural network architecture, users' preferences for each session are represented as an embedding. Based on the user's previous selections and session-level context (e.g., the device used, time), it is possible to predict which songs a user is likely to listen to in the future. Using Variational Autoencoders (VAEs), a multi-round recommender system is presented in [9] which incorporates randomness for fairness throughout and penalizes items based on their historical popularity to promote diversity and minimize bias [13, 10, 12].

Using a bipartite graph, [68] discovers collaborative information using cross-neighbor relation modeling, in which users and items are represented as nodes and their interactions as links. Their high-order collaborative relationships are not only based on the nodes that are directly connected but also on those that are two hops away. User and item dynamics can be captured more accurately by using them along with both user-side and item-side historical sequences. As described in [98], the GLS-GRL system uses item-item co-occurrence graphs to capture user-item inter-

Last-N interactions	[20, 49, 50]
Session-based	[49, 35]
Session-aware	[31, 38, 70, 30, 9, 68, 98] **[87, 88, 89]**

Table 2.2 Sequential recommendations work categorization. The works in **bold** are works presented in this dissertation.

actions over the historical period, as well as item-item co-occurrence graphs to indicate current interactions. GLS-GRL's graph representation learning allows it to obtain long-term and short-term user representations, which can then be merged to produce integrated user representations. By using an interactive constraint-based attention mechanism, relationships between members of a group are encoded into representations of the group that can be used for the recommendation process.

As a whole, the works described previously pertain to single-user recommendation systems. The main focus of this dissertation is to present a framework for achieving group recommendations in a sequential scenario. We present four novel methods for sequential group recommendations, which were initially described in [86, 88]. SDAA takes into account the entire group and dynamically determines a weight according to the members' satisfaction. We utilize this weight to aggregate the following two scores; the average prediction score for an item across all group members and the prediction score of that item for the user most dissatisfied with the previous recommendation round. On the other hand, SIAA evaluates each individual in the group. In every round, it computes a weight per user determined by the user's overall satisfaction and disagreement in the prior round. Finally, Average+ builds on the positive aspects of the standard Average aggregation strategy and aims to mitigate Average's shortcomings in the process. Table 2.2 presents the categorization of the works based on the different approaches to sequential recommendations; Last-N interactions-based recommendations, Session-based recommendations and Session-aware recommendations.

2.4 Reinforcement Learning in Recommendations

In recent years, Reinforcement Learning (RL) has become increasingly popular in recommendation systems [2]. There were some early works in this domain, including [92], in which a web recommender system was proposed. In this model, the state of the environment represents the last N pages visited by a user, actions represent suggestions of pages to the user, and reward represents a weighted sum based on the ranking of the recommended pages and the amount of time the user spent on them. [108] proposes an online personalized news recommendation framework based on DQNs. There are two parts to the framework, offline and online. The offline stage involves training the model, followed by the online stage which involves producing a recommendation and recording feedback from users. In a subsequent stage, the model enters the offline stage and may be modified in response to logged feedback. A long-term reward can be increased by properly modeling the evolving aspects of the news along with the preferences of the users. They also consider user return patterns as part of their exploration strategy for more diverse recommendation options in addition to click/no-click feedback. [36] optimizes recommendation models for long-term accuracy using RL techniques. There are two main areas they consider; cold-start and warm-start. The model was based on the interaction between the environment, i.e., the recommender system, and the agents, i.e., the users. It was, therefore, able to be implemented in environments with insufficient access to content resources.

In [107], the researchers propose a list-wise recommendation framework based on deep RL. By reducing redundant computations in scenarios with large and dynamic item spaces, the method reduces computational costs. The model is trained and evaluated offline, while the recommender system is applied online. A simulation of the user's response (reward) is used to successfully evaluate the model in offline mode The reward is determined by comparing the similarity of the current state with other historical data using a user-agent interaction environment simulator. [106] describes a deep learning movie recommender model that is based on RL, depicting the user's changing interests over time through prioritized experience replay. As part of the recommendation process, agents are used to learn about members' interests and movie features in order to recommend movies based on their preferences.

A user-side sequential music recommendation system is proposed in [56]. This method integrates users' explicit and implicit feedback as part of a Markov Decision Problem (MDP). Users' explicit feedback includes their preferences for music channels, while implicit feedback is generated when new music tracks are requested. [78] also uses an MDP to predict and recommend a new product based on an ordered sequence of choices made by each user. In order to deploy this model readily as an application, several assumptions are made since the e-commerce environment is different from a classical recommendation system.

All the works described above propose solutions to the recommendation task by utilizing RL, but most of them are focused on particular recommendation domains. In this thesis, we present a framework called SQUIRREL, which aims to be more versatile regarding the domains in which it can be utilized and can incorporate a variety of strategies to compensate for the limitations inherent to every recommendation method.

2.5 Explanations for Recommendations

Recommendation methods can be quite complex, and users are unaware of their inner processes. For various reasons, like misinterpreting the collected data, a system may produce false recommendations, i.e., recommendations that are not relevant to a user's interests. As a result, the users may be dissatisfied with the service, which will likely result in a user stopping using the system. One way to address this issue is for the system to provide explanations for its suggestions.

Explanations in recommender systems can have various advantages. First, they provide a level of *transparency*. It is important to explain the reasoning behind a recommendation to ensure the users that the system works as expected and that the suggestions were made for rational reasons. This leads to the second advantage of explanations, *scrutability*. In some cases, explanations can be useful in identifying and correcting the system's misinterpretation of information. As part of the explanation process, the user gains a better understanding of the system and has the opportunity to influence what recommendations are made through the correction of the system's assumptions [80]. This results in the users becoming more trusting in the system, which makes the system more persuasive and effective, resulting in a higher level of satisfaction for the users.

According to how explanations are provided, they can also be classified as user-invoked, automatic, or intelligent [28]. User-invoked explanations are explanations that are provided only after they are requested by the user. Automatic explanations are always produced by the system and cannot be controlled by the user. Finally, intelligent explanations are provided when the system, through some inner process determines that they are required.

An explanation in CF is usually provided as a result of feedback from users. For example, assuming that a user request explanations for a recommendation. In order for this recommendation to be produced, first, a CF system would have to identify users who are similar to the one for whom the explanations are intended. These similar users are often called peers, and the system utilizes them to produce recommendations. In this scenario, the system explains its recommendations, stating that the user is similar to the peers, and the peers rated the suggested item positively [71]. [34] examines how different display styles affect the effectiveness of explanations. An explanation can be presented as either an aggregated histogram of the peers' ratings or as a detailed analysis of their ratings. Additionally, [75] suggests that it is also possible to explain recommendations by showing the user that the recommended items are similar to those they have liked. As a result, many of the items rated highly by the user are provided as an explanation. To investigate the effectiveness of explanations in recommender systems, [93] designed a system to examine the effect of various types of explanations. Overall, this study suggests that appropriately explaining recommendations can benefit recommender systems over a number of different aspects, such as transparency, scrutability, persuasiveness, trustworthiness, and satisfaction.

In recent years, several approaches have been proposed, e.g., [19, 97], that aim to elucidate latent factors utilized by matrix or tensor factorization. [19] provides explanations of recommendations on high-level feature spaces and heterogeneous cross-domain recommendations. [97] propose a multi-task learning solution for providing explanations for recommendations. Two companion learning tasks, one for predicting the relevance of an item to a user and another to predict the opinion that the user would have for that item, are integrated via a joint tensor factorization. Thus, the model estimates not only users' preferences among a list of items, in other words, recommendations, but also users' appreciation of items at the feature level, specifically, opinionated textual explanation.

From another angle, [25] demonstrates how to compute the minimum subsets of user actions that will alter top-ranked recommendations in a counterfactual scenario. The authors propose a provider-side approach to provide concrete explanations for end-users, in which explanations are composed of the bare minimum steps taken by the user that, when deleted, alter the recommendation to an alternative suggestion. Thus, the explanations are short, understandable, and helpful, as they are sets of the least amount of information produced through a counterfactual model using a user's own interaction with the system. Additionally, the explanations do not reveal any personal data about other users, therefore preventing privacy violations by design.

In this thesis, we provide explanations after direct feedback from the user. Comparatively to those works previously mentioned, we provide explanations on why an item or a set of items does not appear in the recommendation list rather than provide explanations on why it does. Furthermore, we are producing the explanations from a system administrator's perspective, rather than appealing to the consumer. This enables us to compose the explanations with more technical details that an average user would not understand but are essential for a system administrator.

2.6 Recommendations in the Health Domain

Health recommender systems are decision-making systems for recommending individualized medical services. A system like this employs a variety of learning algorithms and interprets the patient's health information, often called the patient's profile, to produce healthcare recommendations. Health recommenders can be used in the following scenarios [79]. First, the system is used by a health expert to retrieve information for a medical case. Second, the user of the system is a patient and requires access to reliable and accurate information about their health problems. Finally, in the third scenario, the system proposes combinations of different medications to prevent adverse health effects. This dissertation focuses on the second scenario.

The web is one of the most popular sources of information for patients about their illnesses and possible treatments. However, there are generally two problems associated with the information available on the Internet. First, the information is not always reliable, and second, it is extremely diverse. Users could overcome these problems by using a personalized recommender, which would provide efficient, reliable, and integrated clinical care and research systems, allowing patients to extract relevant information from the enormous amount of heterogeneous information available.

To this end, [62] illustrates what is required of a Health Recommendation System. In particular, the system should be able to cope with imprecise and colloquial terms as well as misspellings. Additionally, the system should be able to handle clinical terms and anticipate when some terms should be negated from consideration. For example, if a patient's health profile states that they do not have cancer, they do not need cancer information. Furthermore, confidentiality must be maintained at all times for the patients.

On the other hand, [76] analyzes the challenges that health recommender systems face. The first challenge is the patients' profiles; the collection and selection of patient health information are complex. Considering the information is provided in a standard format, it has to be organized into a profile for each patient. Due to the fact that health recommender systems accommodate a wide range of user needs, the profile must be tailored to fit each user's needs. The second challenge that health recommender face is trustworthiness. They should be able to present the suggestions in such a way that even if the recommendations are inconvenient, the user can trust the system to guide them in the right direction. Finally, health recommender systems should always be used in conjunction with a health expert in order to use medical assessment and treatment recommendations fully.

Various works are done on health recommendations. [44] proposes an approach to establishing context-aware personalized healthcare recommendation services. [46] assists patients in extracting relevant information from extremely large quantities of heterogeneous and clinical data through semantic annotation of patient profiles and past user queries. In [3], an individualized nutritional recommendation system is proposed based on the health profile of the user and the main guidelines provided by a medical specialist. Additionally, by decoupling users from their properties and items from their properties, [51] defines a collaborative filtering model that assesses the probability that items will be of interest to users matching specific features. Therefore, one may be able to construct a matrix of values indicating the extent to which an item feature affects, in a positive or negative way, an item's usefulness with respect to a specific user property. This would address problems of CF methods, including sparsity, latency, and unfairness against those whose preferences are dissimilar from common preferences.

There are a number of limitations with the recommender systems that have been implemented in the health domain so far, even though they are becoming more popular in the field. To our knowledge, no research has been done in the rapidly evolving area of group recommendations, nor has it been done on incorporating fairness into health care. These new challenges may benefit health experts, and this thesis demonstrates some of these benefits. In order to create a group recommendation list, we have used methods commonly used in pure group recommendation systems.

However, our recommendations have been tailored for the health domain, leveraging the users' semantically annotated Personal Health Records (PHR) to calibrate our recommendations. The result is a direct endorsement of documents relevant to a user based not only on the user's feedback, namely, the ratings that the user gave to items but also on their personalized health profile. In addition, we ensure that all members are treated equally by incorporating the notion of fairness into our model. The importance of this is especially apparent in the health domain, where all group members must be satisfied.

3 SEQUENTIAL GROUP RECOMMENDATIONS

This chapter introduces the concept of sequential group recommendations and analyzes the measures used to achieve these recommendations. In the first section, we describe the problem of sequential group recommendations. Section 3.2 introduces the main methods to identify the user's satisfaction with the system after a series of recommendation rounds. In Section 3.3, we propose three different aggregation functions designed for a sequential group scenario. Finally, Section 3.4 describes the results of the experimentation we performed using real-world datasets.

3.1 Problem Description

In the scope of this work, we consider a system to be a sequential group recommender system when it interacts with a group for a sequence of recommendation rounds. There are two types of groups: stable, whose members stay the same over time, and ephemeral, whose members change between rounds of recommendations. Each round is a complete group recommendation process; first, a single recommendation process is applied to all group members, generating their corresponding prediction lists. To create the group recommendation list, the system then aggregates those prediction lists into one. There are two main objectives that a sequential group recommender system should complete. As a first objective, the items recommended to the group should be as relevant as possible to the group members at each recommendation round. The second objective is that no group members should be discriminated against during a sequence of recommendation rounds.

This is a complex set of objectives that are not easily observed. Group members each have their own likes and dislikes, which can differ greatly from one another. A user is satisfied with a group recommendation list if multiple items are relevant to the user. Take a three-person group as an example. Two of them share the same interests, while the third does not. If we applied a standard group recommendation approach, the third member would most likely be dissatisfied since their interests are dissimilar from the other members, violating the first objective. Additionally, this would continue for all recommendations rounds, thus violating the second objective.

However, we can make a conjecture based on the concept of multiple recommendations. Generally, if a user is unsatisfied with the recommended items at a certain round, they were satisfied at the prior round, or they will be satisfied in the following round. Based on this conjecture, we propose methods to achieve sequential group recommendations that consider how well the system and the group have interacted historically, i.e., what items have been recommended previously and how well the group members have received them.

This sequence of recommendations also generates a need to measure the satisfaction of each user per recommendation round as well as for the entire sequence of rounds. To achieve this, we introduce the notions of satisfaction and disagreement.

3.2 Users Satisfaction and Disagreement Scores

3.2.1 Satisfaction Measure

We need to design a measure that will describe our system's effectiveness per round and achieve the same for numerous rounds in order to evaluate the effectiveness of a sequential group recommender system throughout multiple rounds of recommendations. We introduce the notion of *satisfaction*, expressing how satisfied a user is with the recommendations. This dissertation defines two types of satisfaction: single-user satisfaction and group satisfaction.

3.2.1.1 Single User Satisfaction

Initially, we formally measure how satisfied each user is with the group recommendation Gr_j that they received during the recommendation round j. Specifically, we compare the user's recommendations as an individual, i.e., the predictions they received from a single Recommender System (RS), with the recommendations they receive as a group member. Our system considers the user's predictions as their ground truth, which allows it to view single-user RS as black boxes.

The top-k items for user u, namely the k items with the best prediction scores, are returned as a list called A_u^j. We want to contrast the user's ideal scenario with the items from the group recommendation list. As we take into account each user's top items, rather than the group's top items, we can get a better picture of the users' satisfaction than the average method. This is commonly referred to as the individual loss concerning the group recommendations [58]. Formally:

$$sat(u, Gr_j, j) = \frac{\sum_{i \in Gr_j} p_j(u, i)}{\sum_{i \in A_u^j} p_j(u, i)}. \tag{3.1}$$

where $p_j(u, i)$ returns u's prediction score concerning item i during recommendation round j according to a single RS.

Note that we do not use the scores as they appear in the group list but as they appear in the individual prediction list of the user. Since the aggregation phase of the group recommendation process somewhat distorts the group members' individual opinions, we opt to take into consideration only the personal prediction scores of each group member.

Equation 3.1 describes the satisfaction of a user per recommendation round. To describe a user's overall satisfaction with the entirety of the μ rounds of recommendations we introduce the notion of overall satisfaction.

The overall satisfaction of user u concerning a sequence $\mathcal{GR}$ of μ rounds is the average of the satisfaction scores after each iteration:

$$satO(u, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u, Gr_j, j)}{\mu}. \tag{3.2}$$

3.2.1.2 Group Satisfaction

Having defined each group member's satisfaction score, we can now define the satisfaction score of the entire group. Specifically, we define group G satisfaction concerning a group recommendation list Gr_j as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j, j) = \frac{\sum_{u \in G} sat(u, Gr_j, j)}{|G|}. \tag{3.3}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$, as:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u \in G} satO(u, \mathcal{GR})}{|G|}. \tag{3.4}$$

This measure indicates if the items we report to the group are acceptable to its members. Higher group satisfaction means that the group members are satisfied with the recommendations.

3.2.2 Disagreement Measure

The group satisfaction measure is a necessary tool to identify if our sequential group recommender system is effective. However, there is a flaw in how the group satisfaction score is defined (Equation 3.3). Namely, we take into account the average satisfaction rating for the group. This can cause us to overlook the user's dissatisfaction in some way. Imagine that everyone in the group is extremely satisfied, with the sole exception being a low-satisfied user. The group satisfaction score will then likely remain relatively high, and the user who is least satisfied will usually not be taken into consideration. Thus violating the second objective of a sequential group recommender system. In response to this observation, we establish a new measure, the disagreement score, both at the user and group levels.

User disagreement is defined at the user level as the discrepancy between the user's satisfaction and the highest user satisfaction score across all group members.

$$userDis(u, G, Gr_j, j) = max_{\forall_{u' \in G}} sat(u', Gr_j, j) - sat(u, Gr_j, j). \tag{3.5}$$

This allows us to determine better if a group member is systematically favored (the user will have very low user disagreement scores) or is disregarded (the user has very high user disagreement scores).

At a group level, we intuitively define the group's disagreement as the difference in the satisfaction scores between the most satisfied and the least satisfied member in the group [58].

$$groupDis(G, Gr_j, j) = max_{u \in G} sat(u, Gr_j, j) - min_{u \in G} sat(u, Gr_j, j). \tag{3.6}$$

Subsequently, we define the overall group disagreement of G for the entire recommendation sequence as:

$$groupDisO(G, \mathcal{GR}_{\circlearrowright}) = max_{u \in G} satO(u, \mathcal{GR}_{\circlearrowright}) - min_{u \in G} satO(u, \mathcal{GR}_{\circlearrowright}). \tag{3.7}$$

Ideally, we want this measure to take low values, indicating that the group members are all satisfied to the same degree after a series of recommendation rounds. Higher *groupDis* values will demonstrate that at least one group member is biased against.

3.3 Aggregation Methods for Sequential Group Recommendations

This section outlines three new group recommendation approaches that fulfill the two objectives a sequential group recommendation should complete. They create group recommendation lists with at least one item of relevance to each member of the group, i.e., the group recommendation list is relevant for all group members. Additionally, they consider the previous rounds of recommendations and strive to be fair to all group members throughout a series of recommendation rounds so that no group member is constantly the least satisfied.

Each proposed method is an aggregation method, meaning that a single-user RS has produced a prediction list for each group member. The proposed methods use different mechanics to aggregate these lists into one group recommendation list. The Average and Least Misery are two of the most established aggregation methods used by standard group recommender systems [59]. Average is founded on the idea that every member is regarded as being equal. As a result, the group prediction for an item will be determined by averaging its members' scores.: $avgG(i, G) = \frac{\sum_{u \in G} p(u,i)}{|G|}$, where $p(u, i)$ offers us the user u's prediction score in i (calculated using a single-user recommendation method). In the Least Misery aggregation method, one person has the power to veto any decision made by the group. In this instance, the group prediction score of an item i is the lowest score given to that item in the prediction lists of every group member: $minG(i, G) = min_{u \in G} p(u, i)$.

3.3.1 Sequential Dynamic Adaptation Aggregation Method

The Sequential Dynamic Adaptation Aggregation (SDAA) method builds on the strengths of the Average (equality) and the Least Misery (inclusion of all opinions) aggregation methods and combines them via a weighted summation. In order to calculate an item's prediction score for a group, SDAA uses the following formula:

$$score(G, i, j) = (1 - \alpha_j) * avgG(G, i, j) + \alpha_j * leastG(G, i, j). \tag{3.8}$$

During round j, the $avgG(G, i, j)$ function calculates the average score of item i based on the classic Average aggregation method, while function $leastG(G, i, j)$ provides the least satisfied user's score for i. The *alpha* variable can have values ranging from 0 to 1. If $alpha = 0$, SDAA is transformed into average aggregation, while when $\alpha = 1$, SDAA is transformed into a modified least misery aggregation, which considers only the least satisfied member.

Ideally, this value should fluctuate to better reflect the group's consensus. In each recommendation round, α is dynamically determined by subtracting the group members' minimum satisfaction score from the maximum satisfaction score for the previous round.

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}, j-1) - min_{u \in G} sat(u, Gr_{j-1}, j-1), \tag{3.9}$$

where $j > 1$. When $j = 1$ (the first recommendation round of the system), then $\alpha = 0$, and the aggregation method reverts to that of a classic average aggregation.

Intuitively, suppose the group members are equally satisfied in the last round. In that case, α takes low values, and the aggregation will closely follow that of an average, where everyone is treated equally. On the other hand, if one group member is extremely unsatisfied in a specific round, α takes a high value and promotes that member during the next round.

3.3.2 Sequential Individual Adaptation Aggregation Method

The next proposed aggregation method, Sequential Individual Adaptation Aggregation (SIAA), concentrates on each group member individually. We assign each member a *weight* score. This weight is based on each user's satisfaction and disagreement scores to ensure that the system is calibrated for each user individually.

Specifically, during the aggregation phase, we exploit the concept of overall user satisfaction (Equation 3.2) and user disagreement (Equation 3.5). Each group member is assigned an individual weight, which is applied to an item's prediction score given by the single-user RS (Equation 3.10). The final score of that item for the group is the summation of its weighted prediction scores for all group members.

$$score(G, i, j) = \sum_{u \in G} w_{u,j} * p_j(u, i). \tag{3.10}$$

Users' satisfaction scores can be used to balance recommendations for all group members. Members who were dissatisfied in previous rounds are favored in the following ones. The use of a user's overall satisfaction is a simple approach. Ideally, higher weight should be assigned to users with a low overall satisfaction score. For example, assume the group is in the j-th recommendation round. Then the overall satisfaction score for all previous $j-1$ rounds is given by Equation 3.2. As a result, the weight corresponding to each member at iteration j is:

$$w_{u,j} = 1 - satO(u, \mathcal{GR}_{j-1}). \tag{3.11}$$

where $\mathcal{GR}_{j-1}$ are all previous recommendations at the $j-1$ round of the system.

The users with high overall satisfaction are assigned a lower weight, while those with lower satisfaction scores are assigned a higher one. If the group members are satisfied to the same degree, they are assigned similar weights. Since we compute a user's overall satisfaction score, we ensure that a systematically biased user will have the greatest weight.

However, Equation 3.11 suffers from a drawback. It is slow to compensate for extreme alternations of a user's overall satisfaction throughout the recommendation rounds. For example, suppose a user was highly satisfied in the first n rounds and then extremely dissatisfied in the rest. In that case, the method will require some rounds to assign them a higher weight to promote them. The same is true in the reverse case, where a user was dissatisfied and then highly satisfied. This leads to an aggregation method that promotes users even though they are satisfied during multiple recommendation rounds.

A way to counter this drawback is to shorten the rounds needed for a change in user satisfaction to be reflected in the user's assigned weight. To achieve this, we introduce the *Sequential Individual Adaptation Aggregation method*, or SIAA, that considers a user's overall satisfaction and their disagreement with the previous recommendation round.

$$w_{u,j} = (1-b) * (1 - satO(u, \mathcal{GR}_{j-1})) + b * userDis(u, G, Gr_{j-1}, j-1), \tag{3.12}$$

where b is the weight we use to balance the overall satisfaction and user disagreement scores. Given that we are in the j-th round of the system, $\mathcal{GR}_{j-1}$ expresses the recommendations of all the previous $j-1$ rounds. Finally, Gr_{j-1} consists of the recommendations of the immediately preceding round.

3.3.3 Average+ Aggregation Method

In the following sequential aggregation method, the group recommendation list is considered as a set, unlike the previous method, in which each item was considered individually. Also, the previous group recommendation methods add a single item without considering the existing items in the group recommendation list. Considering that Average aggregation focuses on finding items that satisfy the majority of the group, we propose an aggregation method called Average+ that exploits the Average method and the high group satisfaction scores it produces [86].

Due to its propensity to overlook the outlier in a group, Average also gets high group disagreement scores. When a group member's interests differ from the others, they are usually consistently dissatisfied. This observation prompts us to suggest a new aggregation approach that not only preserves the benefits of the Average method but also lessens the issues caused by the above-mentioned shortcoming.

There are two parts to this aggregation method. We capitalize on the average method's advantages in the first phase and then address its disadvantage in the second. The first phase is simple; we utilize a traditional Average aggregation approach for the group, which yields a lengthy list of candidate items. We retain a much longer list than a typical Group Recommender System (GRS), which often only preserves the k items with the highest score.

Following testing, a list of $5k$ items produces the most successful results, where k is the number of items in the list GRS suggests to the group. This list is denoted as $AvgList_j^G$ for group G at recommendation round j. A longer than $5k$ list typically offers additional items that do not have good enough group prediction scores, indicating that the majority of the group will not find them relevant. Consequently, they cannot be used to benefit from the performance of the average aggregation approach.

Having secured a high group satisfaction, the next phase of the Average+ aggregation method is to minimize group disagreement.

An incremental heuristic method is proposed. There are two steps in this process. First, the item with the highest predicted score in $AvgList_j^G$ is inserted in the group recommendation list Gr_j. The second step of the method is recursive. All the items in the $AvgList_j^G$ are examined individually. The item that is finally selected to be included in the group recommendation list is the one that generates the lowest disagreement score as calculated by the Equation 3.13. This process is continued until all k items have been added to the group's recommendation list.

$$Gr_j = Gr_j \cup \{i \mid min_{\forall i \in AvgList_j^G}(groupDis(G, Gr_j \cup i, j)) \vee i \notin Gr_j\}. \tag{3.13}$$

By incrementally filling the group recommendation list, we can examine all the items and the effects they collectively have on the group.

3.4 Results

For the evaluation section of our work, we did not have access to datasets that contain interactions between groups and a system, where for example, the group as an entity has rated an item. In lieu of such a dataset, we artificially created groups based on information taken from two real datasets, MovieLens, and GoodReads. Since we want to evaluate our proposed aggregation methods for a sequence of recommendations rounds, we need to simulate a time flow, where between the rounds some time has passed. This means that the recommender does not start with all the data available, but the information is sequentially augmented after each round. To simulate this, we first sort each dataset chronologically by the time that each rating was given. Then we split the datasets into chunks, and after each round, a new chunk is introduced to the system.

We also experimented with different group formats. These groups consist of 5 group members and are stable, meaning that the group members remain the same throughout the rounds of recommendation. In order to find similar users we utilized the Pearson Correlation[47] similarity function that produces scores in the range of [-1,1]. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity. We consider two users to be similar if they share a similarity score above 0.5 and dissimilar if their similarity score is below -0.5 The types of groups we are considering are the following:
4 similar – 1 dissimilar (4+1): The four members of the group share similar interests, while the last one does not.
3 similar – 2 similar (3+2): We divide the group into two subgroups. The members of each subgroup are similar to each other, while at the same time, the subgroups are dissimilar to one another, i.e., all members of one subgroup are dissimilar to all

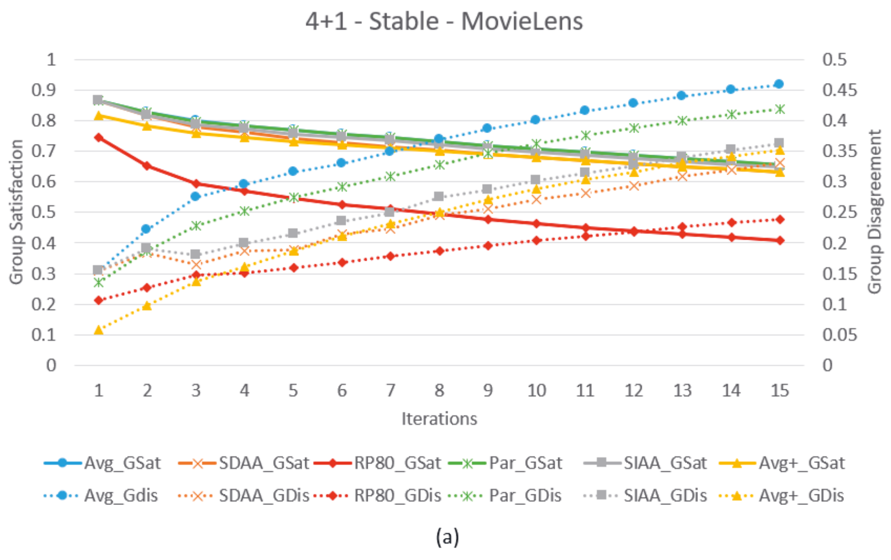


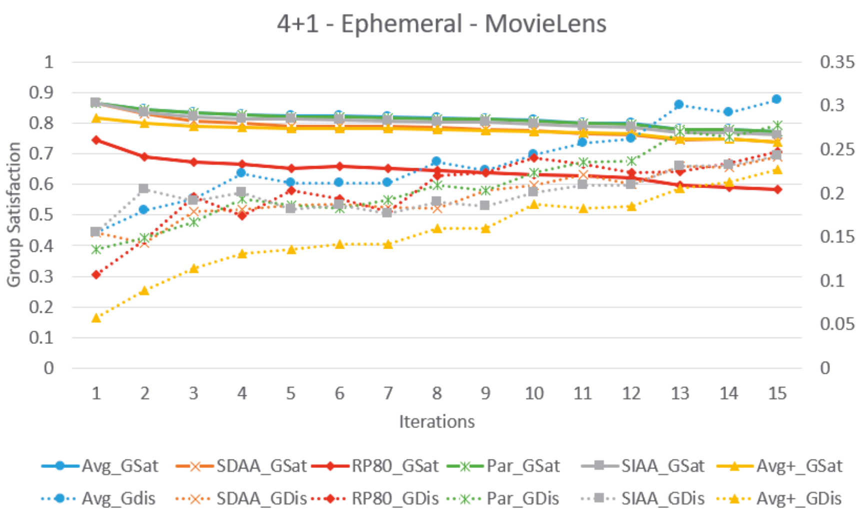
Figure 3.1 Group satisfaction and disagreement scores for 4+1 group format in the MovieLens dataset.

members of the other subgroup.

5 dissimilar: All members of the group are dissimilar from each other.

With these group formats, we want to simulate three different real-life scenarios. First, it's possible that a newcomer's interests will differ significantly from those of the rest of the group when they join an established group for the first time. Consider, for instance, a workplace scenario where a new employee joins a project team that already exists. Second, when two separate groups come together to work on a common project, like two work teams that must cooperate. The last group type mimics a situation in which a group of unrelated individuals gets together for an activity, such as a business lunch or a tour provided by a travel agency.

We compare our methods to three other widely used group recommendation methods; Average, RP80 [4], and Par [99]. For more detailed information on RP80 and Par see Section 4.3.3. Extensive experiments have shown that our proposed methods are overall more effective than the standard group recommendation approaches when applied in a sequential scenario. As shown in Figure 3.1, the SDAA and SIAA methods produce similar group satisfaction scores, with the most effective ones, Average and Par, while simultaneously producing far lower group disagreement scores compared to them, which produce the two highest group disagreement scores. Due to this, our proposed methods outperform the Average method and



(b)

Figure 3.2 Group satisfaction and disagreement scores for 4+1 group format in the MovieLens dataset for ephemeral groups.

fulfill the two main objectives of a sequential group recommender system. They consistently offer relevant items to the group (shown by the high group satisfaction scores) and also ensure that no member is persistently overlooked (shown in the low group disagreement scores).

In comparison to SDAA and SIAA, Average+ shows lower group satisfaction levels. This is to be expected as the extended list produced by Average aggregation serves as the foundation for the group list that Average+ constructs. Average+ investigates items that may yield lower satisfaction but significantly better (i.e., lower) group disagreement scores rather than limit itself to those that would deliver the greatest group satisfaction scores (i.e., the top k returned by Average). As a result, Average+ has lower satisfaction ratings than the Average approach. Similar results can be observed for the alternate group formats and the GoodReads dataset.

Evaluations were also done to examine the performance of the proposed methods for ephemeral groups where the group members change between recommendation rounds. One group member is randomly removed after each recommendation round to mimic the dynamic properties of the ephemeral groups. To maintain the same group size, a new user is randomly added. The newly chosen member is chosen in a way that keeps the group's type constant. For instance, if a 4+1 group has the dissimilar user removed, the new member must be different from the other four group members.

The constant change of group members alters the performance of the proposed methods. However, they still outperform the standard group recommendation approaches. In more detail, as shown in Figure 3.2, Average+ outperforms SDAA in later rounds, despite having lower satisfaction scores in the initial iterations than the other approaches. However, its performance is significantly less than SIAA. SIAA has consistently achieved excellent group satisfaction scores across all test cases, but it can now produce decent disagreement levels with ephemeral groups. This is a result of SIAA's user-centered design philosophy. Compared to an approach like SDAA that is group-focused, it can handle the addition of a new group member better.

Additionally, a discrepancy in SDAA's performance method can be observed. The SDAA technique suffers a negative impact due to the continual member change, with a sharp decline in performance. Every round introduces a new unknown member, forcing the SDAA to employ greater *alpha* values. High *alpha* values have a negative impact on SDAA performance [86] and the value of *alpha* increases to the maximum at each recommendation round because the new member has no prior satisfaction score (Equation 3.9). The traditional Average approach outperforms SDAA because new members are continuously added to the group, preventing SDAA from maintaining its optimal performance.

4 THE SQUIRREL FRAMEWORK

This chapter introduces the SQUIRREL framework – SeQUentIal Recommendations with ReinforcEment Learning, a model based on Machine Learning (ML) techniques in order to enable a sequential group recommender system to select an aggregation strategy automatically. In Section 4.1, we describe the problem, and in Section 4.2, we formalize the SQUIRREL model. In Section 4.3, we detail the various elements of the model. Finally, in Section 4.4, we describe the results of the experimentation process.

4.1 Problem Description

The SQUIRREL model uses Reinforcement Learning (RL) strategies to determine the best group recommendation algorithm depending on the group's present state. The choice to utilize reinforcement learning is intuitive given that the sequential nature of RL closely matches the sequential nature of the sequential group recommendation problem.

The recommendation system should have satisfied all members of the group after a series of recommendation rounds. However, each member's satisfaction changes throughout the rounds. These changes can adversely affect the group recommendation approach that was initially utilized. For example, a group recommendation strategy like the Average works well when all members are similarly satisfied. This is often the case in the first recommendation rounds but is unsuitable when one user is significantly dissatisfied, which may happen in the latter rounds.

As a solution to this problem, the SQUIRREL model defines its environment as the satisfaction of the group members, which changes after each recommendation round. The model observes the current environment and determines which aggregation function should be used to generate the group recommendation list (i.e., an action). For example, if all group members are equally satisfied, the model will select an aggregation function that treats each member equally, such as the classic Average. Alternatively, if a group member has a very low satisfaction score, then the model selects an aggregation function, such as SDAA, that can promote that member and raise their satisfaction score.

4.2 Model Definition

Based on a Markov decision process, the SQUIRREL model engages an agent in interactions with an environment E, in order to maximize an accumulative reward. This process can be described by a tuple of $(S, \mathrm{A}, P_a, R_a)$, where:

- S represents the state of the environment, i.e., the group, and is expressed as the overall satisfaction scores of each member (Equation 3.2). At each round j, we maintain a unique state for each member u of the group.
- A comprises the various aggregation functions used in the SQUIRREL model, where $|\mathrm{A}| = m$. Without any limitations on the number of actions we may add, they can vary from quite basic ones, for example, a standard Average, to considerably more sophisticated ones, such as SDAA.
- $P_a(s, s')$ specifies the likelihood that during round j following the action a, the state s will change to the state s'. Specifically, $P_a(s, s') = Pr(s_{j+1} = s' | s_j = s, a_j = a)$.
- $R_a(s, s')$ is the reward obtained following the transition from state s to state s' under action a. The reward reflects the model's performance. Two reward functions are specified, which use the group members' overall satisfaction (Equation 3.2) and disagreement scores (Equation 3.5). The first method examines the group's overall satisfaction by averaging all members' personal satisfaction scores. The second reward function incorporates the disagreement among group members as well as the overall level of satisfaction.

The model's goal is to seek a policy $\pi(a|s)$ that performs an action $a \in A$ during the state $s \in S$ in an effort to optimize the expected discounted cumulative reward after μ sequential recommendation rounds:

$$max\,\mathbb{E}[R(\mu)] \tag{4.1}$$

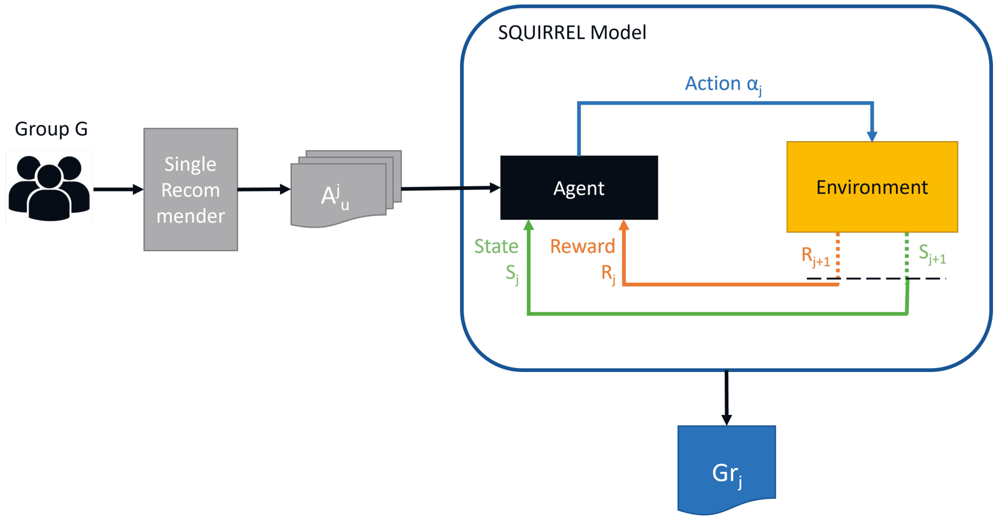


Figure 4.1 The SQUIRREL Model.

where

$$R(\mu) = \sum_{t=0}^{\mu} \gamma R_{\alpha}(s, s') \tag{4.2}$$

with $0 \leq \gamma \leq 1$.

The SQUIRREL model is utilized for a sequence of recommendation rounds to generate a list of suggestions for the group. It is important to remember that each part of the SQUIRREL model may be adjusted and optimized for a particular function. For instance, an application may need to specify an alternative state and/or reward in order to reduce the variations in how users perceive the system's overall performance. On the other hand, alternative actions may be defined by an application that seeks the optimal variable for an aggregation function. As an example, when using the basic Weight Sum aggregation function [94], actions correspond to the various weights.

SQUIRREL, as with any other RL model, is dependent on training. The system requires a vast amount of data during this training phase. As part of training, the model attempts to identify which action (i.e., which aggregation technique) is most efficient given the current environmental state. An RL model must be retrained if any of the model's components, i.e., state, actions, or reward, are modified.

Figure 4.1 describes a SQUIRREL model's recommendation round. At the start of round j, a Recommender System (RS) produces a recommendation list separately for each member of the group noted as A^j_u. In turn, these lists are entered into the

SQUIRREL model, whereby the agent monitors the environmental state S_j, i.e., the current level of satisfaction within the group. Next, it decides on a suitable action α_j for aggregating the lists A_u^j. Consequently, this causes the model's state S_j to transition to the following state S_{j+1}. Additionally, the environment updates by the group members' overall satisfaction, and the reward R_{j+1} is calculated. Lastly, the model provides the group with the resulting recommendation list Gr_j.

4.3 The SQUIRREL Model

This Section describes in depth the SQUIRREL model's three main elements – state, action, and reward. These components are adaptable and may be tailored to the user's demands. In this thesis, the model uses measures mentioned in Chapter 3. Specifically, the users' overall satisfaction defines the model's state. The group satisfaction and disagreement scores express the reward functions. Finally, the model's actions include various state-of-the-art aggregation techniques in addition to the previously suggested aggregation methods, SDAA, SIAA, and Average+.

4.3.1 State Definition

The specification of the environmental state is a critical component of the SQUIRREL model. This state is what characterizes the group members' present status. Therefore, we require a state that places an individual emphasis on each group member and how satisfied they are with the recommendations they had received in previous rounds of recommendations. This will prevent the model from mistakenly ignoring any group member, which is likely to happen if we consider the group as a whole.

Therefore, we define our model's state as an array containing the overall satisfaction of each group member. Following each recommendation round, the model's internal state is updated with the group members' new total satisfaction scores. The model maintains the satisfaction of each group member through the most recent round of suggestions.

4.3.2 Reward Definition

The only means by which the model has to decide if an action it selected was suitable is the reward provided by that action. Based on what the model tries to achieve, the reward function may be defined in a variety of ways. In this thesis, two different reward functions are proposed.

The first reward function utilizes the group satisfaction score (Equation 3.3), which describes how well the group members received the recommended items. The satisfaction score will reflect how successfully the system can balance the group members' desires. A high group satisfaction indicates that the system successfully located items relevant to most group members. On the other hand, a lower group satisfaction score denotes a failure of the system to meet its objectives.

Formally,

$$R_s(\mathcal{GR}^j) = groupSatO(\mathcal{GR}^j) \tag{4.3}$$

where $\mathcal{GR}^j$ refers to all the rounds up to the jth one.

However, according to the previous discussion in Chapter 3, a sequential group recommender system needs to complete two objectives; propose items relevant to the group members and keep all the members equally satisfied. The above-mentioned reward function can only achieve the first objective. With that in mind, we also define a second reward function that can address both objectives by considering the overall group satisfaction and the group disagreement (Equation 3.6) scores.

As a second alternative reward of the SQUIRREL model, we can utilize the harmonic mean of group satisfaction (Equation 3.3) and group disagreement scores (Equation 3.7), namely their F-score. Considering the input functions that F-score needs, we use $1 - groupDisO$ to simulate the group agreement.

$$R_{sd}(\mathcal{GR}_j) = 2\frac{groupSatO(\mathcal{GR}_j) * (1 - groupDisO(\mathcal{GR}_j))}{groupSatO(\mathcal{GR}_j) + (1 - groupDisO(\mathcal{GR}_j))}. \tag{4.4}$$

Overall, this strategy consists of two components, *groupSatO*, and *groupDisO*, reflecting the degree to which an item is preferred by the members of the group and the level at which the members disagree or agree with each other and aiming for an appropriate balance of these components.

4.3.3 Actions Definition

The actions are the driving force behind our model. The agent observes the state of the environment and the history of the rewards it has already achieved and decides to apply an action. The action will generate changes to the state, enabling it to calculate a reward. The actions are the obvious choice to carry out the group recommendation process since they are the only method that can result in changes in the model.

In this thesis, an RS generates a recommendation list for each group member. This process is considered a black box. These separate recommendation lists can then be consolidated into one group recommendation list as part of the group recommendation process.

SQUIRREL utilizes the following six aggregation methods.

1. **Average.** As the name implies, the Average Aggregation method consists of averaging the predicted scores for each item among the members of the group. This implies that each member's predicted score of an item is treated equally.
2. **RP80 [4].** The relevance and disagreement scores of groups are combined in this method. Group relevance, $avg(i, G)$, is defined as the average prediction score among all group members of group G. In this work, the group disagreement, $dis(i, G)$, is calculated as the average pair-wise disagreement between the predicted scores of each group member for item i. $dis(i, G) = \frac{2}{|G|(|G|-1)} \sum(|p(u, i) - p(u', i)|)$, where $u, u' \in G$, and $u \neq u'$. The Average Pair-wise Disagreement indicates how closely group members agree on the relevance of the data item i. Accordingly, the total score for item i based on the group G is as follows: $\mathcal{RP}80(i, G) = (1 - w) * avg(i, G) + w * (1 - dis(i, G))$, where w is a tuning factor for the group relevance and disagreement.
3. **Par [99].** This method incrementally adds items with the highest combination of Social Welfare (SW) and Fairness (F) scores to the group recommendation list. An item's Social Welfare is calculated as the average level of satisfaction among its members. Similarly to this thesis, [99] calculates satisfaction using equation 3.3 when there is only one item, i, in the group recommendation list. Consequently, $SW(i, G) = groupSat(i)$. The variance over members' satisfaction scores is employed as Fairness, $F(i, G)$. For each item, the final score is calculated as: $\mathcal{PAR}(i, G) = \lambda * SW(i, G) + (1 - \lambda) * F(i, G)$.
4. **SDAA.** The Sequential Dynamic Adaptation Aggregation method, at round

j, calculates a weight w that is equal to the difference between the most satisfied and least satisfied group members from the previous round of recommendations, $w^j_{SDAA} = max_{u \in G} sat(u, Gr_{j-1}) - min_{u \in G} sat(u, Gr_{j-1})$. At the start of round j, a weighted sum of the average score for the item in the group and the least satisfied user's predicted score for the item will be used to compute the final group prediction score: $score(G, i, j) = (1 - w^j_{SDAA}) * avgG(G, i, j) + w^j_{SDAA} * leastG(G, i, j)$. For item, i at round j, $avgG(G, i, j)$ returns the average score across all members of the group, while $leastG(G, i, j)$ returns the least satisfied user's score for i. SDAA becomes a simple average aggregation function when $j = 1$, i.e. when the first round of recommendations is performed. A more detailed discussion can be found in Chapter 3.3.1.

5. **SIAA**. As with SDAA, Sequential Individual Adaptation Aggregation also uses weights for aggregation. SIAA focuses on each member of the group individually, unlike SDAA, which considers the group as a whole. The following formula determines a weight that balances a member's overall satisfaction and the user disagreement from the preceding recommendation round: $w^j_{SIAA}(u) = (1 - b) * (1 - satO(u, \mathcal{GR}_{j-1})) + b * userDis(u, Gr_{j-1})$, with b being the weight for balancing overall satisfaction and user disagreement. During the j-th round of recommendations, $\mathcal{GR}_{j-1}$ represents all previous $j - 1$ recommendations. After completing the single-user RS process, this weight is applied directly to each item's predicted score. For an item, the predicted group score is the average of the weighted scores of all group members. A more detailed discussion can be found in Chapter 3.3.2.

6. **Average+**. The group satisfaction scores, produced by the classic Average aggregation function, are among the highest. However, the disagreement score is among the worst. The Average+ aggregation method is proposed to address this drawback, which consists of two phases. As a first step, an Average aggregation is used. A second phase in the process involves iteratively populating the recommendation list with items that yield the lowest score achievable for group disagreement: $Gr_j = Gr_j \cup \{i \mid min_{\forall i \in AvgList^j_G}(groupDis(Gr_j \cup i)) \vee i \notin Gr_j\}$, where $AvgList^j_G$ represents the list of the top k items following the first phase of the aggregation. A more detailed discussion can be found in Chapter 3.3.3.

It is worth mentioning that although in this thesis, only these aggregation functions are examined as actions, the model can be augmented with additional aggregation methods.

4.4 Results

To evaluate our model, we utilized the datasets mentioned in Chapter 3.4, MovieLens and GoodReads, and an additional dataset, namely the Amazon dataset. We also kept some of the group formats as in Chapter 3.4, the **4+1**, and **5 Dissimilar** group formats. For each of these group types and each dataset, we formulated 100 different groups. We used 80% of the groups for each group format as the training set and the remaining 20% for the testing phase.

Aside from the previously mentioned group formats, we also take into account an additional scenario to evaluate the model. When the model is trained for only one type of group, it will be restricted to learning the format associated with that group only. Nevertheless, such an approach does have some merit. An application can, for example, group people randomly for an activity in order to facilitate more varied social interactions. This situation necessitates using a system capable of accommodating the diverse needs of individuals who are different in nature. The SQUIRREL model should, however, be used in more universal circumstances. In order to accomplish this, 50% of each group type's training sets and ten (10) groups obtained from their respective test sets were randomly selected to form an additional testing scenario.

In order to evaluate the model, both reward functions mentioned in Chapter 4.3.2, R_s and R_{sd} were used. When SQUIRREL is presented with either reward function, it chooses the optimal aggregation based on what maximizes that specific reward function. The same behavior can be observed for all three datasets used in the experiment. The model can also identify when an otherwise effective aggregation method is not optimal for a round. SDAA and SIAA, for example, lack the necessary information when they formulate their first recommendation. As a default, SDAA uses a classical average, while SIAA utilizes the previous round's disagreement, a factor that is not present in the first round. Subsequently, both SDAA and SIAA then resort to traditional Average aggregation. In contrast, Average+ does not possess any such disadvantage. The model can detect this in the first round of recommendations and select the most effective aggregation function.

In order to demonstrate the overall effectiveness of each group recommendation

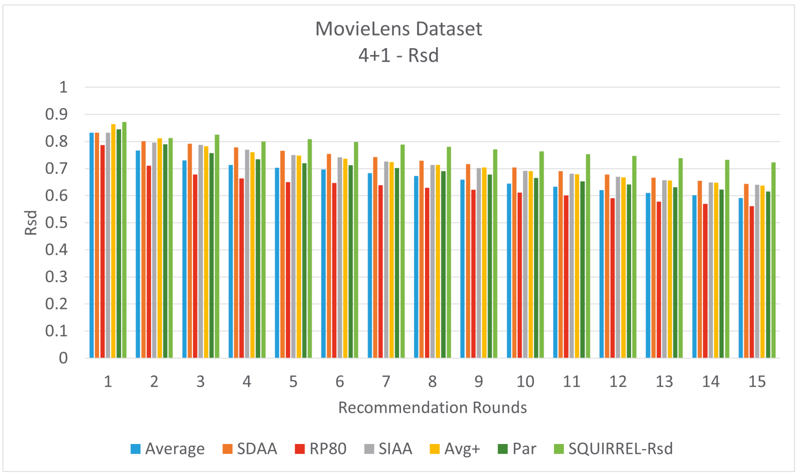


Figure 4.2 R_{sd} scores for 4+1 group format in the MovieLens dataset.

approach, it is better to use the reward function R_{sd} that combines the group satisfaction and group disagreement scores. Figures 4.2, 4.3 and 4.4 show the R_{sd} scores for the 4+1 group format for each dataset, MovieLens, GoodReads and Amazon. For the other group formats, similar results are obtained.

In general, when considering both objectives for the sequential group recommender system, namely high group satisfaction and low group disagreement scores, the SQUIRREL model offers the best results, regardless of which reward function was utilized. However, as is expected, the reward function R_s, which only factors in the overall satisfaction of the group members, produces slightly better satisfaction scores. On the other hand, the R_{sd} reward function that also considers the group members' disagreement produces significantly lower group disagreement scores.

Overall, the SQUIRREL model offers the best group satisfaction scores. For the initial round of recommendations, the model decides on one aggregation function that maximizes the selected reward function. However, the model has some flexibility in the selection of that function. As a result of choosing a different aggregation method in the initial round, the results for the following rounds are impacted as well. Both group satisfaction and disagreement improve due to the slight change in the first round. Additionally, SQUIRREL only produces the second-best group disagreement score behind RP80 because RP80 was explicitly designed to generate low disagreement scores. However, because of this specialization of the RP80 method, it also produces very low group satisfaction scores.

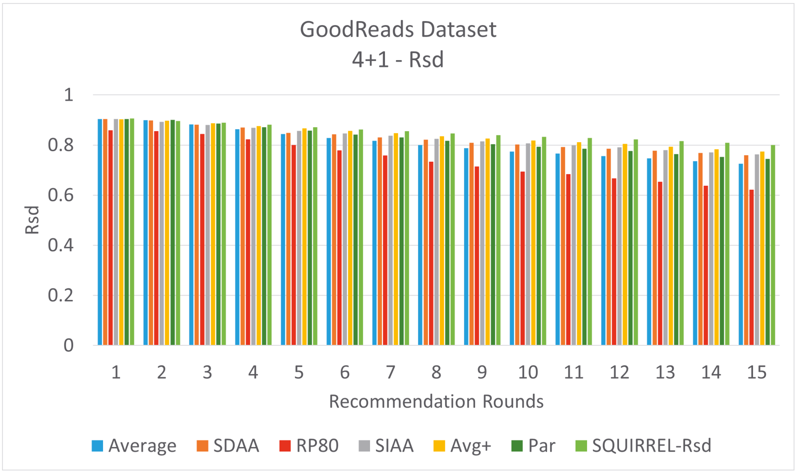


Figure 4.3 R_{sd} scores for 4+1 group format in the GoodReads dataset.

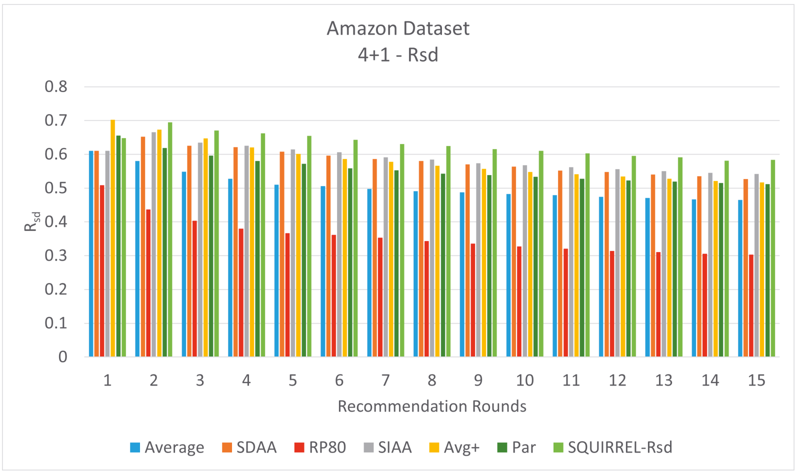


Figure 4.4 R_{sd} scores for 4+1 group format in the Amazon dataset.

There is a difference in the behavior of the aggregation methods among the three datasets when both group satisfaction and disagreement scores are considered, namely the R_{sd} scores. SDAA is the best-performing one for MovieLens (Figure 4.2), Average+ is the best-performing one for GoodReads (Figure 4.3), and SIAA is the best for Amazon (Figure 4.4). Discrepancies in performance among the three datasets can be attributed to their sparsity; in particular, the GoodReads and Amazon datasets have a higher sparsity than the MovieLens dataset. Consequently, the RS is adversely affected, and the number of highly relevant items for each member of the group is reduced. In contrast, the single RS generates a greater number of relevant items per group member in the MovieLens dataset, which is the densest of the three datasets. The SDAA aggregation benefits from this since it ensures a balance between members' satisfaction and disagreements. SDAA is more beneficial when group members share common relevant items.

Conversely, the sparsity of the GoodReads and Amazon datasets is favorable for Average+ and SIAA aggregation methods. As the number of items with great relevance per user is lower, Average+ will consider the best possible items for the group during its first phase of selecting the items with the highest average score. Afterward, Average+ can choose the items that generate the least disagreement among the group. In SIAA, scores are only calculated based on the satisfaction and disagreement of each individual member, not the group's; thus, sparse datasets can be handled more effectively. In comparison to all other individual aggregation methods, the SQUIRREL model can adapt to different datasets and outperform them all.

Finally, the quality of the recommendations produced by the SQUIRREL model is evaluated by calculating the Normalized Discounted Cumulative Gain (NDCG) [39], whereby the best recommendations should appear at the top of the group recommendations list:
$NDCG(u, G, j) = \frac{DCG(u,G,j)}{IDCG(u,G)}$, with $DCG(u, G, j) = \sum_{i \in Gr_j} \frac{|i \cap A_u^j|}{log_2(rk(i,Gr_j)+1)}$,
where $rk(i, Gr_j)$ is the rank of item i in the group recommendation list Gr_j. For u, IDCG represents the best possible outcome. The results from NDCG are supplemented with the Discounted First Hit (DFH) metric, which measures whether a user has seen a highly relevant item in the early ranks of the group recommendation list: $DFH(u, Gr_j) = \frac{1}{log_2(fh+1)}$, where fh refers to the rank of the first item on both the group recommendation list as well as the user's own prediction list produced by a single RS. Due to the fact that NDCG and DFH refer to one user, the group

score is determined by calculating them separately for each group member and then averaging over all members. The results are computed at the end of each round.

With either reward function, the SQUIRREL model has the two highest NDCG values, which means it can provide more useful items to groups. Moreover, it recommends highly rated items by the users based on the DFH scores. On the other hand, the Average+ aggregation method shows a decrease in NDCG scores since it takes into account items that aren't always of great interest to each member. High NDCG scores are generally associated with aggregation methods such as Average, SDAA, SIAA, Par, and SQUIRREL, which produce higher satisfaction scores. Because the group satisfaction score and NDCG scores convey the same basic concept regarding the importance of items in the group recommendation list to each user, this situation is not surprising.

5 WHY-NOT QUESTIONS AND EXPLANATIONS

In this chapter, we introduce the concept of Why-not questions and explanations and how they can provide a system administrator with valuable information in order to adjust the various variables of a recommender system. Section 5.1 presents the problem of Why-not questions and explanations, and Section 5.2 briefly describes the recommendation model and its corresponding variables. Section 5.3 introduces the characteristics of a Why-not question, while Section 5.4 describes the explanations provided for the Why-not questions. Finally, Section 5.5 highlights the results of the evaluation process of the proposed model.

5.1 Problem Description

The problem of explaining why a recommendation was made is well documented, with explanations and recommendations presented in conjunction. End-users and system designers can thus gain insight into why a certain item is recommended. The explanations can diversify in their presentation or granularity, depending on the consumer, specifically either the recommender's user or the system's designer. This work introduces the notion of *Why-not* questions. When asking why-not questions, the purpose is not to determine why particular items were included in the results but rather to determine why they were not included. Additionally, this work is based on the concept of *post-hoc*, model-based explanations for Why-not questions. These explanations are produced after completing the recommendation process and are based on the RS's model.

Particularly for system administrators and engineers, Why-not questions are essential since they are required to understand the system better and to get indications on how to debug it. Consider, for instance, an e-shop that offers suggestions for products to customers. Imagine a system administrator has discovered that a specific company's products are never presented to customers. Consequently, the engineer may need to identify the reason for the situation and devise an alternative(s) to rectify it. This could benefit the final user in terms of a wide range of recommendations or even help promote a specific company's products, which are not included in the system's suggestions. This work assumes the explanations are consumed by the system designer, which, for simplicity, will be referred to as the user of the system.

5.2 Recommendation Model

A user-based Collaborative Filtering RS produces recommendations that require explanations. The first step of this process is to compute similarities between the users. Based on two users' ratings, $sim(u, u')$ is used to count their similarities. When $sim(u, u')$ exceeds a threshold value th, and the users have rated more than $numI$ common items, a user u' is considered similar to a user u. Further refinement is achieved by retaining only $numP$ users with the highest similarity scores, denoted as *peers* of u, $Peers_u$. For any item i that u has not yet rated, a prediction score $p(u, i)$ will be predicted based on the peers of u. In order to avoid recommending items only liked by one (or a few) peers and unknown to others, items are recommended when more than $numPI$ peers have rated them. Furthermore, the recommendation list A_u that the RS produces consists of the k items with the highest prediction scores. Finally, the notation $pos_{R_u,i}$ is used for the index of the item i in R_u.

5.3 Why-Not Questions

In lieu of explaining all items proposed by the system, the user asks questions instead. Suppose a user is unhappy with the results provided by a movie recommender system. Then, they can pose questions like *Why were no dramas recommended?* The system will provide answers based on the characteristics of the system and the information regarding these items.

Three main characteristics distinguish why-not questions: absenteeism, granularity, and dependency on existing recommended items. Absenteeism is easily determined from false negative results, i.e., the items that should have been reported (in a particular position) but were not. Granularity refers to the grouping of missing items. The dependency reflects scenarios in which an item that appears in the recommendations (true positive) and an item that does not (false negative) should appear

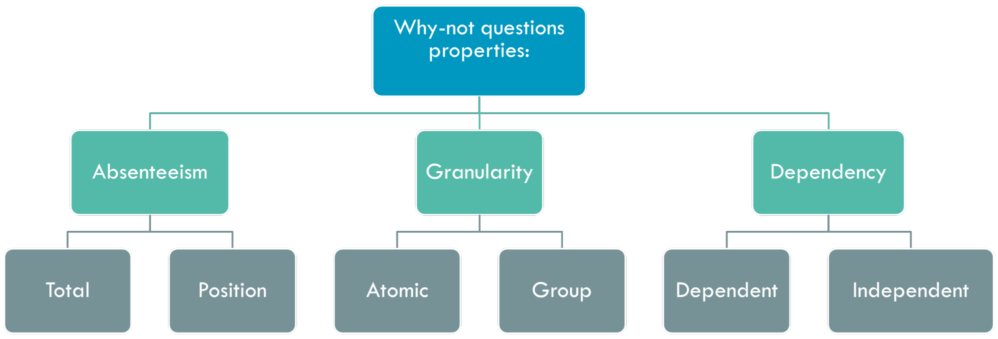


Figure 5.1 Properties of Why-not Questions.

together in the recommendation list.

In more detail, *absenteeism*-based why-not questions are distinguished between *total* absenteeism and *position* absenteeism. As an example, questions such as *Why not Matrix?* can be considered total cases since they refer to items that do not appear on the recommendation list and do not have a specific requirement for placement in that list. Questions that are concerned with item ranking, such as *Why should Matrix not be ranked first?*, are position-based. The granularity feature, which refers to the degree of detail of the question, is divided into atomic cases and group cases. Users may ask questions about particular items (atomic case), such as: *Why not Matrix?*, as well as about groups of items with similar characteristics (group case), such as: *Why not comedies?* Finally, dependency takes into account items that are usually returned together in answers or should be returned in a specified order. For instance, assume the questions *Why are there not any comedies, but there are dramas?* Figure 5.1 showcase all the properties that a Why-not Question can have.

The formal definition of Why-not questions is the following. Consider a set of items I, a user u, and u's recommended items $A_u / subseteq I$ produced by an RS. Why-not questions take the form of:

$$wn = \{(m, pos, d) \mid m \in I \; and \; pos \in \{1, \ldots, |A_u|\} \; and \; d \in A_u\} \tag{5.1}$$

Equation 5.1 is flexible enough to account for all cases mentioned previously, i.e., absenteeism (m and pos), granularity, and dependency (d). Despite not being explicitly apparent, a *granularity* Why-not question can be derived by expanding the group to the related items in I. For example, a Why-not question of the style *Why not dramas?* can be represented by $wn = \{(Titanic,,), (TheGodfather,,)\}$, given that the system can find these two movies in its database.

As a further step, Equation 5.1 is used to express Why-not questions' different properties (absenteeism, granularity, dependency). A Why-not question always contains the *absenteeism* property, so both subcategories (total and position) are discussed when they also concern granularity and dependency. For the sake of simplicity, only the notation for a set of items is given (the group subcategory of the granularity property). However, the formalization of the Why-not questions is not affected because both granularity questions can be expressed using a set format (i.e., an item is part of a set that consists only of a single item). Each case of the Why-not question is described intuitively, accompanied by examples from a movie recommender system, and then formalized for that case.

- **Total Absenteeism:**
  - *Independent*: The user asks why some items do not appear in the recommendation list.
    Example-Atomic: *Why Matrix is not there?*
    Example-Group: *Why are there not any dramas?*
    Formally, an independent total absenteeism Why-not question is:

    $$wn_{ti} = \{(m,,) \mid m \in I \setminus A_u\}$$

  - *Dependent*: The user asks why certain items do not exist while others (that usually appear together) exist.
    Example-Atomic: *Why Matrix is not there but Terminator is?*
    Example-Group: *Why there are not any thrillers when there are action films?*
    Formally, a dependent total absenteeism Why-not question has the form:

    $$wn_{td} = \{(m,,d) \mid m \in I \setminus A_u \text{ and } d \in A_u\}$$

- **Position Absenteeism:**

- *Independent*: The user can ask about the ranking of a set.
  Example-Atomic: *Why is Matrix not ranked first?*
  Example-Group: *Why are dramas not in a higher position?*
  Formally, an independent position absenteeism Why-not question takes the form:

$$wn_{pi} = \{(m, pos,) \mid m \in A_u \ and \ pos_{A_u,m} < pos\}$$

- *Dependent*: The user asks why certain items do not appear higher in the recommendation list than other recommended items.
  Example-Atomic: *Why not place Matrix before Terminator?*
  Example-Group: *Why not place comedies before dramas?*
  Formally, an independent position absenteeism Why-not question is:

$$wn_{pd} = \{(m, pos, d) \mid m \in I \ and \ pos > pos_{A_u,d} \ and \ d \in A_u\}$$

5.4 Why-Not Explanations

The system seeks to provide meaningful explanations to the system designers when answering a Why-not question. In order to be meaningful, the information should be adequate to help the designer understand why the items are not recommended as expected. For this reason, the input is split into distinct components that can explain - individually or combined - the Why-not question provided by the user. These components are the input item set, the sets of all and the similar users given the user in question, the set of ranking scores, and the recommender system design (hyperparameters). To accommodate the different sources of error, a multi-type structure, called an *explanation*, is defined as follows:

A *Why-not explanation* for a Why-not question on the recommendations of a user *u* is a set of parameters of the recommender system responsible for the absence of the missing item(s) from the (specific positions of the) recommendation list.

The explanations are distinguished between *general* explanations, which can appear in any recommender system, and *model-specific* explanations that are based on the inherent parameters of the CF recommendation model.

5.4.1 General Explanations

It is possible that the items that appear in a Why-not question do not exist in the system's database. Then the explanation is straightforward; the system does not suggest this item since it is an unknown. Another explanation can be drawn from the number of reported top-*k* items. If *k* reflects only a small number of items, it is possible that the recommendation list includes the missing item, although it is ranked lower. This explanation is only applicable for the first 2*k* items in the list provided by the RS. For items further down that list, we consider other model-specific explanations. Finally, different items may be scored the same way by the system. It uses a specific method in order to break ties, for example, placing the first encountered item in the database at the top. The system may label the tie-breaking method as the culprit when a user asks why an item has been neglected.

5.4.2 CF Explanations

In Collaborative Filtering (CF), items are recommended to a user based on what similar users have liked in the past. It is, therefore, likely that all possible explanations will revolve around the peers of the user. Occasionally, an item remains invisible to the system because no peers have given it a rating. If no user has ever rated that item before, the system will also not recognize it. Besides these two explanations, an answer to a question such as "Why not item A?" is determined by combining the results for the three questions : (i) what is the number of peers that have rated it, (ii) what scores they have assigned to it, and (iii) what is the similarity between them and the user. Peer ratings are ignored when there are only a few (less than three) to prevent false suggestions. The system scores items low when all or most peers have assigned them a low rating. Additionally, the degree to which a peer is similar to the user is fundamental. In the case of high-similarity peers, a peer's dislike of an item will influence the item's predicted score.

The answers to the three questions above are represented in the form of explanation tuples (*peer, score, similarity*). Three values in each tuple describe a peer who has rated an item: (i) the peer's id, (ii) their rating of the item, and (iii) their level of similarity with the user.

The system rechecks the same information whenever the user inquires about an item's ranking in a recommendation list. It provides answers to questions such as:

"Why was not item A ranked higher?" by providing an analysis of the item's statistics: the number of peers who have rated the item, how favorable they rated it, and the degree of similarity between these peers and the user. There is a vagueness to this question in that it questions the general ranking of an item without comparing it to any other item; the question is independent. This is treated as if it were a total absenteeism-based question by the system. A second solution is to arbitrarily select one item in the list to transform the independent questions into dependent ones. In this case, the selection is based on the specifics of the question - whether it is higher or lower rank. For instance, a question such as "Why was not item A ranked higher?" can be rephrased into "Why not place item A before item B?".

Providing explanations for dependent Why-not questions, such as "Why not item A but item B?" or "Why not place item A before item B?", presents a greater degree of difficulty as it involves several items, with some existing, some not, combining explanations and Why-not explanations. An analysis of the process that produces an explanation for the first question is provided. A similar process can produce explanations for the second question. First, two separate questions are posed in response to the Why-not question. The first part relates to the user's question: "Why not item A". The second question is "Why not item B". Due to the fact that the system has promoted item B over item A (either by excluding A from the recommendation list for *total* Why-not questions or by ranking item B higher for *position* Why-not questions), B should have more explanation tuples than A and B's tuples should also have higher values than A's tuples. The combination of the answers to these two questions produces the final explanation.

In the case of a user expressing a *group* Why-not question, for example, "Why not more dramas?", the system provides an explanation that incorporates the answers it would have provided separately for each item. For each item in the same category as the Why-not question, the system formulates a Why-not question, provided a peer has previously rated that item A user-friendly output can be created by distilling the results into something that can be consumed easily by the user. For example, "Your peers prefer dramas, but they have not rated the same movie", indicating that the peers prefer dramas, but each rated a different film. It is for this reason that none of them were suggested.

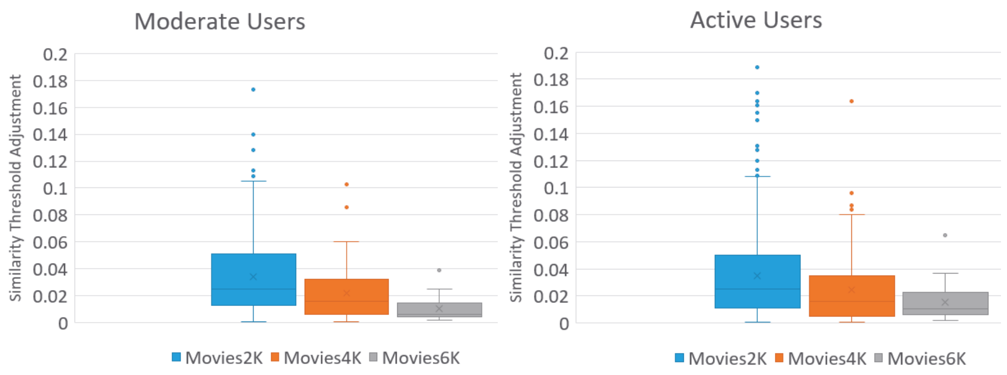
5.5 Results

The experimental evaluation used the MovieLens dataset. Several parameters were tested regarding the proposed model, specifically the users' attributes for whom the Why-not question was posed and the popularity of the missing movie. In order to analyze the behavior of the algorithm two test sets were generated. First, 100 users were randomly selected who have rated a few items (45 to 55), denoted as *Moderate Users*, and second, another 100 randomly selected users who have rated a large number of items (145 to 155), named *Active Users*. Additionally, a variety of popular movies were chosen for the study. Randomly selected 4 sets of 100 movies with 2K, 4K, 6K, and 8K ratings, respectively, were used.

Many explanations can be derived from CF variables listed in Section 5.2. The target item is not included in the recommendation list as a result of these variables. The *k* explanation indicates that the item in question appeared further down the recommendation list after the top-k items. The *numP* explanation means that a raise to the *numP* threshold needs to occur in order to find enough peers that have rated that specific item. The *Peers* explanation is displayed when no peer has rated the item. Finally, *Tuples* refers to explanations that include peer information.

In the Moderate User case, less than 20% of movies with 2K ratings could be explained by peer information. On the other hand, more than 95% of movies with 8K ratings could be explained by peer information. Additionally, over half of the movies with 2k ratings had no ratings from the user's peers, whereas this percentage has dropped to almost zero for movies with 8k ratings. The Active Users case shows similar numbers.

Based on the comparison of the outputs for the two sets of users, Active Users have a higher number of explanations about the top peers not rating an item *numP*) than Moderate Users. Users with more ratings are more likely to have numerous similar other users. As the number of peers used is not a percentage but a constant, there is a higher probability that the selected users have not rated the films in question. Moreover, for the Active Users case, explanations due to the fact that no peers had rated the item were fewer compared to corresponding ones for the Moderate Users. Again, this is due to the higher number of peers among Active Users. A demonstration of how a system administrator could use the model to rebalance the variables of the CF system was also provided. A list containing all of the movies that



(b)

Figure 5.2 The similarity threshold (*th*) adjustment needed for the recommender to be able to calculate a preference score for the missing items corresponding to a *Peers* explanation for (a) moderate, and (b) active users.

have not been rated by any of the user's peers (corresponding to the *Peers* explanations) was generated. An examination of all the users in the system was conducted in order to find the new similarity threshold the recommender needs, in order to calculate a prediction score for an item in that list. A comparison was then made between the threshold used initially and the newly calculated threshold as shown in Figure 5.2. Both Moderate, Figure 5.2a, and Active Users, Figure 5.2b, require only small adjustments in the similarity thresholds. This is a demonstration of how a system administrator can take advantage of the explanations for debugging their system.

6 MULTIDIMENSIONAL GROUP RECOMMENDATIONS IN THE HEALTH DOMAIN

This chapter introduces a group recommendation model focused on achieving fair recommendations for a group of patients. We exploit not only classic recommendation input, such as ratings of documents but also the health profiles of the users. A new similarity measure based on these health profiles is proposed. Section 6.1 describes the problem, while Section 6.2 analyzes the proposed similarity function. Section 6.3 describes the group recommendation model utilized, and Section 6.4 describes the process of creating a health-focused document corpus with the corresponding ratings given by the users. Finally, Section 6.5 analyzes the results of the evaluation process.

6.1 Problem Description

Nowadays, health information is increasingly being searched on the web. While there is a great deal of information available online, the quality and quantity of that information often hinder interested patients from obtaining the correct information. A reliable solution is for patients to be led by healthcare providers to trustworthy data sources on the Web [6]. However, health providers do not always have much time to dedicate to their patients, which makes guiding them in the correct way a difficult task.

Moreover, it has been demonstrated that utilizing group dynamics to alter behavior is highly effective in enhancing social support, e.g., by promoting cohesion among participants in physical activities [14] and lowering relapse rates in smoking [21]. As part of these therapy sessions, caregivers provide their patients with access to more appropriate online resources. However, finding information online for a group of participants is more difficult than finding online information for a single patient. Even though health professionals usually address closely related health conditions, education, health literacy, and psychoemotional status play a significant role in determining the content that the health professional should suggest and the recommendations should be based upon the aforementioned factors.

To achieve this goal, this thesis proposes the development of a new semantic similarity measure that integrates the patient's medical problems, educational level, health literacy, and psycho-emotional status. Moreover, this thesis investigates the use of collaborative filtering to develop a multidimensional group recommendation model in the health domain.

6.2 Users Similarity based on Health Profile

Generally, diverse information can be used to find similarities between users. First, there are the ratings given by the users to the documents. Second, to capture patient problems, group members fill out particular questionnaires (i.e., the ALGA-C questionnaire [45]). An ontology is then used to model and store all captured information. According to the answers to these questionnaires, specific values are automatically calculated and stored in the patient's profile in relation to the key areas of their profile. Among others, scores exist for health literacy, educational level, cognitive closure, and anxiety. Considering that each source yields different knowledge, four similarity measures are defined. These measures are then combined to calculate the similarity between two users.

Similarity based on ratings. A user is deemed similar to another if they have similarly rated data items, i.e., their interests in health documents are similar. Any similarity function that can compare two sets of ratings corresponding to two users can be utilized. Such a similarity function is Pearson correlation [47], a fast and efficient method that is suited to collaborative filtering. The Pearson correlation coefficient is calculated directly between two users, with a score ranging from -1 to 1.

$$RatS(u, u') = \frac{\sum_{i \in X} (r(u,i) - \mu_u)(r(u',i) - \mu_{u'})}{\sqrt{\sum_{i \in X} (r(u,i) - \mu_u)^2} \sqrt{\sum_{i \in X} (r(u',i) - \mu_{u'})^2}} \tag{6.1}$$

where $I(u)$ denotes all the items that user u has rated, $X = I(u) \cap I(u')$, μ_u is the mean of the ratings in $I(u)$. Finally, $r(u,i)$ is the rating that u has given to item i.

Similarity based on health information. If two people have similar health problems, they usually have similar interests in health-related documents. Using the International Statistical Classification of Diseases and Related Health Problems[1] (ICD10), we are able to record health problems and then identify similarities between users utilizing this standard medical classification list maintained by the World Health Organization.

The ICD10 taxonomy is represented as a tree, where the nodes represent health problems. The taxonomy is structured in such a way that only one path connects two individual nodes (acyclic). The structure also features significant similarities between sibling nodes at lower levels compared to those at higher levels. In light of the discrepancy between the similarity of the health problems at different levels, nodes are weighted in accordance with their level. Nodes at different levels are differentiated based on these weights. There should be more similarity between siblings nodes in the higher levels than those in the lowest levels (root node). To that end, we assign a weight to each node based on the level it resides at.

Specifically, let A be a node in the ontology tree. Then,

$$weight(A) = w * 2^{maxLevel-level(A)} \tag{6.2}$$

where w is a constant, $maxLevel$ is the maximum level of the tree, and $level(A)$ is a function that returns the level of each node.

In addition, let $anc(A)$ be the direct ancestor of A. Intuitively, we need to consider the distance between two nodes and the level that those nodes belong. To achieve that, we use the concept of the Lowest Common Ancestor (LCA) of two nodes A and B, defined as the lowest node that has both A and B as descendants, where each node can be a descendant of itself.

Then, for computing the distance between A and B, we compute their distance from $LCA(A, B)$. We first identify the path that connects A (and respectively B) with $LCA(A, B)$. Specifically, given that $LCA(A, B) = C$, $path(A, C)$ returns a set of nodes including A, its direct ancestor $anc(A)$, its direct ancestor $anc(anc(A))$, and so on, until C is reached, without including C in the set. The distance between A and C is

[1] /urlhttp://www.icd10data.com/

calculated by accumulating the weights of each node in the path as:

$$dist(A, C) = \sum_{n \in path(A,C)} weight(n) \tag{6.3}$$

In conclusion, we employ the following formula to determine the similarity of two nodes, A and B.

$$simN(A, B) = 1 - \frac{dist(A, C) + dist(B, C)}{maxPath * 2} \tag{6.4}$$

Note that we divide the sum of the two distances with $maxPath * 2$ to normalize the overall similarity so that the function *simN* returns a value in the range of [0,1]. We define *maxPath* as follows:

Let T be a tree, and A and B two nodes in T, with A being a node in the highest level and B the root. Then,

$$maxPath = dist(A, B) \tag{6.5}$$

We can calculate the similarity between two health problems using the measures described above. However, users typically have more than one health problem in their profile.

Let $Problems(u)$ be the list of health problems of user $u \in U$. Given two users, u and u', we calculate their overall similarity by considering all possible pairs of health problems between them. Specifically, we take all the problems in $Problems(u)$ and calculate the similarity with all the problems in $Problems(u')$. For each distinct problem from u, we consider only the health problem of u' with maximum similarity. Overall, the semantic similarity between u and u' is defined as:

$$SemS(u, u') = \frac{\sum_{i \in Problems(u)} ps(i, u')}{|Problems(u)|} \tag{6.6}$$

where

$$ps(i, u') = max(\forall_{j \in Problems(u')} \{simN(i, j)\}) \tag{6.7}$$

Similarity based on education and health literacy level. There is a wide range of complexity and depth in how online health sources present a problem. A source that is in accordance with a user's level of health literacy and education will be more attractive to them. Patients with a low health literacy score, for example, may not be interested in a document describing their health problem in great detail but will appreciate a document that provides clear guidance on how to manage it. The first document would be much more appealing to a patient with a high literacy score.

It is not uncommon for health documents containing similar information to be of interest to people who have similar educational and health literacy levels. Therefore, the Euclidean distance between these two values is used to measure the similarity between them.

$$EducStatusS(u,u') = 1 - \frac{\sqrt{(HLit(u) - HLit(u'))^2 + (EducLvl(u) - EducLvl(u'))^2}}{\sqrt{2 * maxDif^2}} \tag{6.8}$$

Where $HLit(u)$ is a function that returns the health literacy level of user u and $Educlvl(u)$ is a function that returns their education level. To better combine these scores with the ratings and health problems similarity scores, we normalize them so that the function returns values in the $[0,1]$ range. The variable $maxDif$ represents the maximum difference between the two education or health literacy scores. Finally, since we want the similarity score, not the distance between the users, we subtract the distance score from 1.

Similarity based on psycho-emotional status. Finally, anxiety and cognitive closure affect the documents people prefer in specific periods, as anxiety and cognitive closure can fluctuate over time. As such, we use the Euclidean distance between the values of those two properties. As psychoemotional questionnaires are being answered periodically, we consider only the latest measurements on these each time.

$$PhychStatusS(u,u') = 1 - \frac{\sqrt{(Anxiety(u) - Anxiety(u'))^2 + (CognCl(u) - CognCl(u'))^2}}{\sqrt{2 * maxDif^2}} \tag{6.9}$$

where $Anxiety(u)$ is a function that returns the user's anxiety level, and $CognCl(u)$ is a function that returns their cognitive closure status. In the same manner, as the similarity method based on education and health literacy levels, the Euclidean distance is normalized in the $[0,1]$ range and subtracted from 1 to provide a similarity ranking.

Similarity between users. A total similarity score is calculated by combining the different values obtained from the different methods of computing similarity scores. Accordingly, we assign weights to each similarity score that determine its significance as we argue that not all perspectives of information are equally important.

$$S(u,u') = \alpha * RatS(u,u') + \beta * SemS(u,u') + \gamma * EducStatusS(u,u') + \delta * PhychStatusS(u,u') \tag{6.10}$$

where $\alpha + \beta + \gamma + \delta = 1$.

6.3 Group Recommendation Model

We exploit the Collaborative Filtering (CF) recommendation model to generate lists of relevant items for the group members. To find the similar peers required by the CF, we utilize the above-mentioned similarity function (Equation 6.10).

We propose a new aggregation method, called *AccScores* method, in which we accumulate the items' scores. We add the scores as they appear in each group member's individual prediction list, A_u as produced by a single RS, in a set called *accDoc*. The first item we select to include in the group recommendation list, *Gr* is the one with the highest score in *accDoc*. After each selection, we update a helper structure *accUser*, consisting of the users and their accumulating prediction scores. If a user u has a lower score than the rest in the *accUser* structure, for the next choice, we will select an item that exists in the A_u and simultaneously has the highest possible score in the *accDoc*. If many users have the same lowest score, we select the user chosen the least number of times.

Having constructed a group recommendation list and given a set of recommendations for a group to its caregiver, there may be a patient u who is the least satisfied in the group, i.e., all items are irrelevant to them. Meaning the group recommendation list is not fair for u. In real-life situations, the caregiver is responsible for the needs of all group members, and the recommender should make sure that the documents proposed are relevant and fair to all group members. The degree of fairness for a patient u given a set of recommendations Gr is defined as follows: $fairness(u, Gr) = \frac{|X|}{|Gr|}$ where $X = A_u \cap Gr$.

We define *group discord* as the difference between the maximum and minimum fairness in the group in order to determine the cohesion of the group and to determine if any group member is biased against.

$$groupDiscord(G, Gr) = max_{u \in G} fairness(u, Gr) - min_{u \in G} fairness(u, Gr) \qquad (6.11)$$

Group discord should take lower values, as that would indicate that the group members are equally treated. A high score suggests that one of the group members is not as satisfied as the other members.

6.4 Dataset Creation

To evaluate the proposed model, we will need patients' profiles that contain their health problems and the corresponding ratings they have given to documents. However, the personal profiles of these patients are not publicly available. Due to a number of factors, among them ethical and legal constraints, the acquisition and usage of such personal information is restricted.

We generated a synthetic dataset to compensate for this limitation. Initially, we utilized publicly available 10.000 chimeric patient profiles [40]. These profiles include information equivalent to that found in a medical database. Among other details, we analyze the details of the patient's admission, demographics, socioeconomic data, labs, and medications. Furthermore, the profiles rely on the ICD10[2] ontology to define each patient's health problems, rendering this dataset perfect for accommodating the semantic similarity method.

Utilizing these profiles, we generate a synthetic dataset consisting of a document corpus and user ratings. In particular:

- Document Corpus
  - *Create document corpus.* Several documents were generated per node in the second level of the ICD10 ontology tree.
  - *Assignment of Education and Health Literacy Levels.* Using percentage scores corresponding to the five different education levels, we divide the documents into subgroups and assign them education levels. A document's education and health literacy scores cannot be vastly different. Documents with high education levels are unlikely to be useful to users

[2] http://www.icd10data.com/

with low literacy levels. It is also unlikely that a document with a high level of health literacy will also have a low level of education. Thus, each document will be assigned a health literacy score that is the same, one higher or one lower than its educational level.

- Rating Dataset
  - *Divide the patients into groups.* It is assumed that each patient has rated several documents. We divide patients into three sets based on the number of ratings they have given, *occasional*, *regular*, and *dedicated*. Each group of users gave *few*, *average*, and *a lot* of ratings, respectively.
  - *Assignment of Education and Health Literacy Levels.* Assigning education and health literacy levels to patients follows the same procedure as assigning them to documents, which is described above.
  - *Assignment of Anxiety and Cognitive Closure.* As anxiety and cognitive closure tend to change rapidly, these scores are regularly measured. As a result, only recent data is considered in our methods. Each patient in our dataset receives one score for anxiety and cognitive closure. Similar to the method used to determine the level of education and health literacy, five percentages are used to divide the patients. The score for cognitive closure will be determined by anxiety. A person's anxiety regarding their health problems increases the need for them to understand them.
  - *Simulate a power law rating distribution.* Power law distributions are generally used to rank documents based on user preferences. To demonstrate this, several documents were randomly selected and considered the most popular.
  - *Generate documents to rate.* For each patient, the ratings they would give to documents that had a direct bearing on their health would be distinguished from those that were irrelevant to their health. We assigned ratings to both groups of documents based on the assumption that patients would be interested in both.
  - *Generate ratings.* Finally, a 1-to-5 rating was assigned randomly to each item generated above.

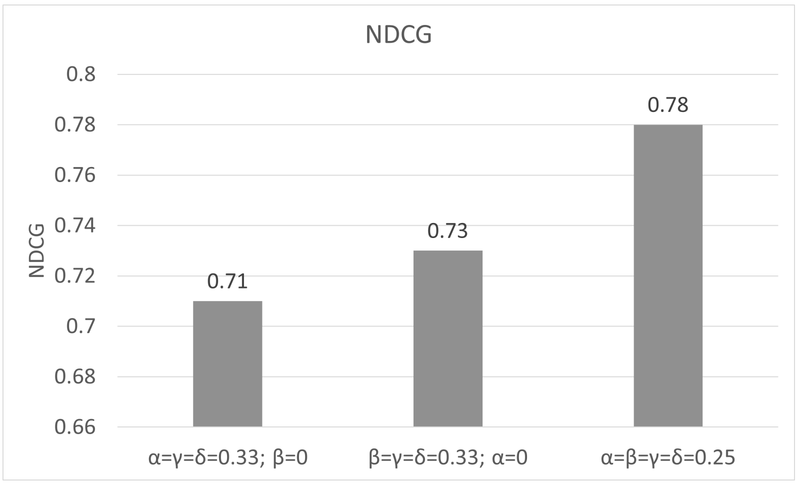


Figure 6.1 NDCG values for different values of α, β, γ and δ.

6.5 Results

We examined the recommendations for individual users to compare the similarity functions. A total of 50 users were used, and 20 percent of their ratings were hidden. We then applied our method with different values for α, β, γ and δ and predicted a score for the hidden items. Note that α represents the weight assigned to rating similarity *RatS*, β to health problems similarity *SemS*, γ to education/health literacy similarity *EducStatusS* and δ to anxiety/cognitive closure similarity *PhychStatusS*. To evaluate the similarity functions, we used the normalized Discounted Cumulative Gain [39] where the relevance of the items appearing in a user's recommendation list is correlated to the relevance of the items in an ideal scenario. The hidden items provided the ground truth in computing the NDCG scores.

As shown in Figure 6.1, compared to the *RatS* similarity function, *SemS* provides better results. Further analysis was performed on how *PhychStatusS* and *EducStatusS* affect them. Despite introducing the two new similarities, *SemS* still gives better results. However, combining all of the similarity functions yields the best results. By combining the similarities of *SemS* and *RatS*, they can compensate for each other's shortcomings. *SemS* is capable of identifying patients with similar health problems, meaning that they are interested in documents concerning the same health issue. *RatS* finds all the other patients who have similar interests in documents but do not necessarily have similar health problems. Adding *EducStatusS* and *PhychStatusS* to the equation further improves the results. The peer selection process is further refined when they are considered, which ensures that the recommendations are more accurate.

We compare our proposed group recommendation approach, AccScores, to three other aggregation approaches. The classic Average, Borda [22], and Fair [84] aggregation methods. In the Borda method, each document receives one point for placing last in the list rankings, two points for placing second to last, and so on, culminating in k points for placing first. An item's total points are accumulated from all group members' individual prediction lists, A_u. Documents with the highest total points are ranked first on the group recommendation list, Gr. The item with the second highest number of points is assigned the second position up to the best k items.

The Fair aggregation method takes into account pairs of patients within a group in order to generate recommendations. Specifically, a document i is added to the group recommendation list, Gr, if for patients $u, u' \in G$, $i \in A_u \bigcap A_{u'}$, $i \notin Gr$, and i is the highest ranking document in $A_{u'}$. Suppose k, i.e., the length of the group recommendation list, is higher than the number of documents we can obtain from the method outlined above. In that case, additional documents are added to the list by iterating through the A_u lists of the group members and including in each iteration the document with the highest ranking not already included in the group recommendation list.

To evaluate the methods, 40 groups were randomly selected with approximately the same degree of similarity. Each aggregation method's results were evaluated by measuring the average distance between the *top-k* recommendation list generated for the group and the list generated for each individual user. The Kendall tau distance was used to calculate the distance between two ranking lists, which is a measure of the number of pairwise disagreements between the lists [42]. Formally, $K(t_1, t_2) = |\{(i,j) : i < j, (t_1(i) < t_1(j) \wedge t_2(i) > t_2(j)) \vee (t_1(i) > t_i(j) \wedge t_2(i) < t_2(j))\}|$, where $t_1(i)$ and $t_2(i)$ are the rankings of the element i in t_1 and t_2 lists, respectively.

There are very few differences between the *AccScores* aggregation method and the other methods, although *AccScores* outperforms the rest as illustrated in Figure 6.2. The group members are all similar due to the case study, so when aggregating their *top-k* recommendations, the group is presented with the items that are most relevant to the group, regardless of the aggregation design.

A far more interesting topic of evaluation is the individual satisfaction of each

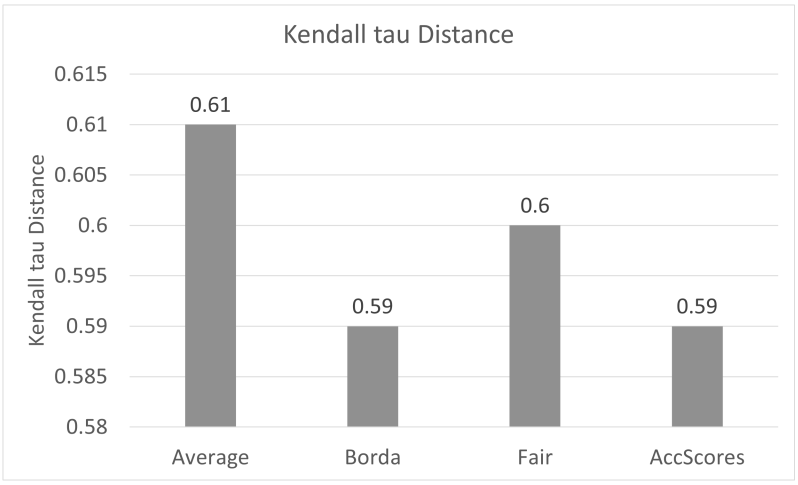


Figure 6.2 The Kendal tau Distances between the users' prediction lists and the group recommendation lists.

group member. Our goal is to understand how the recommendation list impacts each individual member of the group by calculating the *group discord*. All members of the group should be treated equally in order to ensure fairness. It is essential that the system does not return a biased result against a member since accurate information about people's health is crucial. Therefore, a method that generates lower *group discord* values is more suitable for our purposes. As shown in Figure 6.3, *AccScores* generates the lowest group discord scores, demonstrating the method's advantages. Essentially, the *AccScores* method identifies nearly equally fair items for every member.

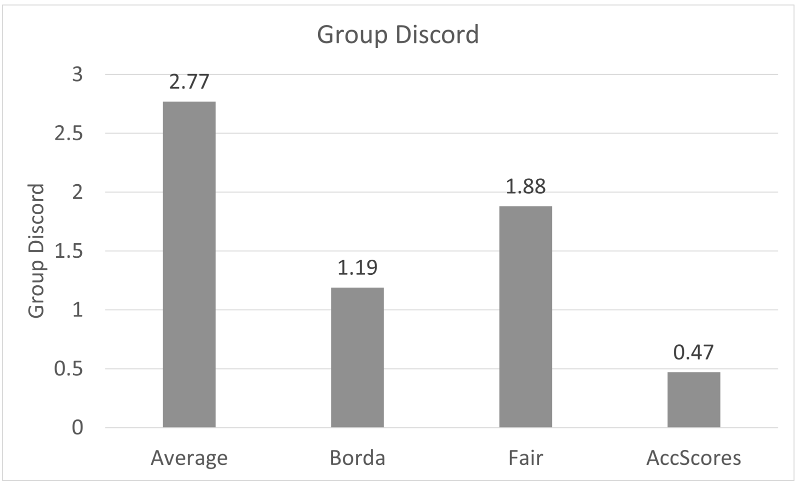


Figure 6.3 Group Discord scores for 40 groups.

7 CONCLUSIONS

7.1 Summary of Contributions

One of the most important applications of recommender systems is group recommendations, where a group of users interacts with the system. When designing a group recommender system, it is imperative to consider the members' diverse interests in order to provide suggestions that are relevant to the entire group. Additionally, the system would need to consider the previous interactions between it and the group before offering suggestions. The system should consider a sequence of previous interactions and how the members responded to earlier recommendations and adjust accordingly. As recommender systems directly affect the users' experience, responsibility constraints should be incorporated into their design. Such responsibility constraints can include, for example, *fairness*, where all group members should be treated equally, and *transparency*, where the system provides explanations for its recommendations. This thesis primarily explores how recommender systems can be designed to incorporate these responsibility constraints.

Specifically, the first contribution of this dissertation, discussed in Chapter 3, answers the first research question *RQ1. How to define sequential group recommendations and why are they needed? How to design group recommendation methods based on them?*. We propose three methods, SDAA, SIAA, and Average+, that aim to provide fair recommendations in a sequential group recommendation scenario. SDAA considers the group members' overall satisfaction with the previous rounds of recommendations. If a member is considerably less satisfied than the rest, the member's interests are prioritized. SIAA assigns a weight to each group member based on their satisfaction in the previous recommendation round. Finally, Average+ is an extension of the classic Average aggregation method, where instead of just considering the average score for all members, it also takes into account the members' satisfaction with the entire list. Experiments on two real-world datasets show that our proposed methods are more effective than other state-of-the-art group recommendation techniques when they are applied in a sequential scenario. More specifically, SDAA and Average+ are the two methods that yield the best results on the MovieLens dataset. Average+ is the most effective method on the GoodReads dataset regardless of the group format. The Average+ aggregation method is the most efficient when ephemeral groups are considered.

The second contribution addresses the second research question *RQ2. How to exploit reinforcement learning to select a group recommendation method when the system's environment changes after each recommendation round?* and is analyzed in Chapter 4. It extends the work done in Chapter 3 by proposing the SQUIRREL model which is based on reinforcement learning for attaining fairness in sequential group recommendations. As the state of the model, we consider the satisfaction of the group members and, as actions, various group recommendation methods. We evaluate our model using two reward functions, one based on the average satisfaction of the members and the second based on the satisfaction and disagreement in the group. We evaluate the model with three real-world datasets and show that it compensates better than stand-alone group recommendation methods when the group information rapidly changes. In more detail, regardless of the group format and the dataset, SQUIRREL has the best performance. Both reward functions produce high-quality results. Group disagreement scores are slightly higher when the reward function focuses only on user satisfaction rather than considering it in combination with group disagreement scores.

The third contribution of this thesis focuses on transparency and answers the third research question *RQ3. How to design questions and produce explanations for why a set of items did not appear in a recommendation list or at a particular position?* in Chapter 5. In particular, the use of "Why-not" questions for recommender systems is explored, where a user asks the system why an item did not appear in the recommendation list. We formally define a Why-not question and propose a model for providing explanations to such questions for a Collaborative Filtering (CF) recommender system. We evaluate our model using a real-world dataset and demonstrate how a system administrator can utilize the provided explanations to calibrate a CF model.

The final contribution is outlined in Chapter 6 and answers the fourth research question *RQ4. How to incorporate various health-related aspects in group recommenda-*

tions?. We propose a group recommendation model designed primarily for the health domain. We consider various factors related to the users' health, including, but not limited to, their health problems and their interests regarding health documents. The main focus of this work is to provide fair recommendations to a group. Due to the patient's sensitive health information, we also created two synthetic datasets in order to evaluate our model. The evaluation demonstrates that the proposed model is more effective than standard group recommendation strategies. In more detail, the proposed similarity function improves upon existing similarity functions by considering the user's various health aspects and ratings. Moreover, the proposed group recommendation method achieves far greater fairness than other group recommendation models while providing similar high-quality results.

7.2 Future Work

Several research areas may be explored as a result of this dissertation. The most notable research topics are the following. We want to combine the fairness and transparency constraints in a sequential group recommender system, where the system would also provide the group with an explanation of why a suggestion was made. This has the potential to be complex since the system considers multiple recommendation rounds, and such explanations can be very cumbersome for an average user to understand. So a degree of generality in the explanation details should be considered.

Additionally, we intend to continue developing the SQUIRREL model to extend the action set beyond aggregation techniques. However, this is a complex change, and various aspects of the model will need to be adjusted. Many complex methods of group recommendations require alternate input forms, such as graphs, to function. By its nature, SQUIRREL requires an extensive training phase, and including more complex methods will increase the training time. Subsequently, we must implement various optimizations to our model before considering additional actions.

More straightforwardly, one can test different reward functions in the SQUIRREL model. This can also be combined with the various fairness definitions in the literature. An advantage of this is that observing how each fairness approach behaves under the same test conditions would be possible. It would also be interesting to see if there is any correlation between the fairness definition and the group recommendation methods. As an example, let's take the following. If the reward is just the average satisfaction of the group members, then an aggregation method that is very likely to be selected is the Average since it is closely related to average satisfaction. However, such a correlation between more complex group recommendation methods and reward functions is not readily apparent.

An additional avenue of research is the explanations. We want to generalize our Why-not explanations to be model-agnostic. Our proposed method is based on a user-based CF recommendation model, and the explanations we provide reflect that by including various CF variables. We want to generalize our explanations to such Why-not questions so that they can be used across multiple recommender systems.

Another aspect of explanations we consider is counterfactual explanations for group recommender systems. The counterfactual explanations consist of interactions between a group of users and the system, such as ratings or reviews, that are responsible for a particular item being recommended. In other words, if these interactions were altered or removed, the system would have recommended another item. An example of such research can be found in [25]. However, it was only for single recommender systems. In the future, we would also like to expand this to include group recommendations. Due to the complexity of group recommendation methods, such an approach must be very efficient since the possible combinations of all group members' interactions with the system are vast.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". In: *IEEE Trans. Knowl. Data Eng.* 17.6 (2005), pp. 734–749.

[2] M. Mehdi Afsar, Trafford Crump, and Behrouz Far. *Reinforcement learning based recommender systems: A survey*. 2021. arXiv: 2101.06286 [cs.IR].

[3] Giuseppe Agapito et al. "DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion". In: *2th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2016, New York, NY, USA, October 17-19, 2016*. 2016, pp. 1–8. DOI: 10.1109/WiMOB.2016.7763190. URL: https://doi.org/10.1109/WiMOB.2016.7763190.

[4] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. "Group Recommendation: Semantics and Efficiency". In: *PVLDB* 2.1 (2009), pp. 754–765.

[5] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. "Group recommendations with rank aggregation and collaborative filtering". In: *RecSys*. 2010.

[6] Gina M. Berg et al. "Evaluating the quality of online information about concussions". In: *Journal of the American Academy of PAs* 27 (2014), pp. 1547–1896. DOI: 10.1097/01.JAA.0000442712.05009.b1. URL: http://journals.lww.com/jaapa/Fulltext/2014/02000/Evaluating%5C_the%5C_quality%5C_of%5C_online%5C_information%5C_about.15.aspx.

[7] Alex Beutel et al. "Fairness in Recommendation Ranking through Pairwise Comparisons". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019. ISBN: 9781450362016.

DOI: 10.1145/3292500.3330745. URL: https://doi.org/10.1145/3292500.3330745.

[8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. “Recommender systems survey”. In: *Knowledge-Based Systems* 46 (2013), pp. 109–132. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2013.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0950705113001044.

[9] Rodrigo Borges and Kostas Stefanidis. “Enhancing Long Term Fairness in Recommendations with Variational Autoencoders”. In: *MEDES*. 2019.

[10] Rodrigo Borges and Kostas Stefanidis. “F2VAE: a framework for mitigating user unfairness in recommendation systems”. In: *SAC*. 2022.

[11] Rodrigo Borges and Kostas Stefanidis. “Feature-blind fairness in collaborative filtering recommender systems”. In: *Knowledge and Information Systems* 64.4 (2022), pp. 943–962. ISSN: 0219-3116. DOI: 10.1007/s10115-022-01656-x. URL: https://doi.org/10.1007/s10115-022-01656-x.

[12] Rodrigo Borges and Kostas Stefanidis. “On Measuring Popularity Bias in Collaborative Filtering Data”. In: *EDBT/ICDT Workshops*. 2020.

[13] Rodrigo Borges and Kostas Stefanidis. “On mitigating popularity bias in recommendations via variational autoencoders”. In: *SAC*. 2021.

[14] Irwin Brandon, Kurz Daniel, Chalin Patrice, and Thompson Nicholas. “Testing the Efficacy of OurSpace, a Brief, Group Dynamics-Based Physical Activity Intervention: A Randomized Controlled Trial”. In: *J Med Internet Res* 18.4 (May 2016), e87.

[15] John S. Breese, David Heckerman, and Carl Myers Kadie. “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”. In: *UAI*. 1998.

[16] Robin Burke. “Hybrid recommender systems: Survey and experiments”. In: *User modeling and user-adapted interaction* 12 (2002), pp. 331–370.

[17] Da Cao et al. “Attentive Group Recommendation”. In: *SIGIR*. 2018.

[18] Shuo Chang, F. Maxwell Harper, and Loren Gilbert Terveen. “Crowd-Based Personalized Natural Language Explanations for Recommendations”. In: *RecSys*. 2016.

[19] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. "Learning to Rank Features for Recommendation over Multiple Categories". In: *ACM SIGIR*. 2016.

[20] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. "Where You Like to Go Next: Successive Point-of-Interest Recommendation". In: *International Joint Conference on Artificial Intelligence*. 2013.

[21] Derek Yee Tak Cheung et al. "Using WhatsApp and Facebook Online Social Groups for Smoking Relapse Prevention for Recent Quitters: A Pilot Pragmatic Cluster Randomized Controlled Trial". In: *J Med Internet Res* 17.10 (Oct. 2015), e238.

[22] Peter Emerson. "The original Borda count and partial voting". In: *Social Choice and Welfare* 40.2 (2013), pp. 353–358. DOI: 10.1007/s00355-011-0603-9. URL: https://doi.org/10.1007/s00355-011-0603-9.

[23] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. "Personalized News Recommendation with Context Trees". In: *CoRR* abs/1303.0665 (2013).

[24] Yingqiang Ge et al. "Towards Long-Term Fairness in Recommendation". In: WSDM '21. Virtual Event, Israel: Association for Computing Machinery, 2021. ISBN: 9781450382977. DOI: 10.1145/3437963.3441824. URL: https://doi.org/10.1145/3437963.3441824.

[25] Azin Ghazimatin, Oana Denisa Balalau, Rishiraj Saha Roy, and Gerhard Weikum. "PRINCE: Provider-side Interpretability with Counterfactual Explanations in Recommender Systems". In: *WSDM*. 2020.

[26] Giorgos Giannopoulos, George Papastefanatos, Dimitris Sacharidis, and Kostas Stefanidis. "Interactivity, Fairness and Explanations in Recommendations". In: *Adjunct Publication of the 29th ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2021*. 2021, pp. 157–161.

[27] Elizabeth Gómez, Ludovico Boratto, and Maria Salamó. "Provider fairness across continents in collaborative recommender systems". In: *Information Processing & Management* 59.1 (2022), p. 102719. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2021.102719. URL: https://www.sciencedirect.com/science/article/pii/S030645732100203X.

[28] M Sinan Gönül, Dilek Önkal, and Michael Lawrence. “The effects of structural characteristics of explanations on use of a DSS”. In: *Decision support systems* 42.3 (2006), pp. 1481–1493.

[29] Francesca Guzzi, Francesco Ricci, and Robin Burke. “Interactive Multi-party Critiquing for Group Recommendation”. In: *RecSys*. 2011.

[30] Casper Hansen et al. “Contextual and Sequential User Embeddings for Large-Scale Music Recommendation”. In: *RecSys*. 2020.

[31] Negar Hariri, Bamshad Mobasher, and Robin Burke. “Context-aware Music Recommendation Based on Latenttopic Sequential Patterns”. In: *RecSys*. 2012.

[32] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2015), 19:1–19:19.

[33] Ruining He and Julian McAuley. “Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering”. In: *WWW*. 2016.

[34] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. “Explaining collaborative filtering recommendations”. In: *CSCW*. 2000.

[35] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. “Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations”. In: *RecSys*. 2016.

[36] Liwei Huang et al. “A deep reinforcement learning based long-term recommender system”. In: *Knowledge-Based Systems* (2021). ISSN: 0950-7051.

[37] Anthony Jameson and Barry Smyth. “Recommendation to Groups”. In: *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer-Verlag, 2007, pp. 596–627. ISBN: 9783540720782.

[38] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. “Adaptation and Evaluation of Recommendations for Short-term Shopping Goals”. In: *RecSys*. 2015.

[39] Kalervo Järvelin and Jaana Kekäläinen. “IR Evaluation Methods for Retrieving Highly Relevant Documents”. In: *SIGIR Forum* 51.2 (Aug. 2017), pp. 243–250. ISSN: 0163-5840. DOI: 10.1145/3130348.3130374. URL: https://doi.org/10.1145/3130348.3130374.

[40] Uri Kartoun. “A Methodology to Generate Virtual Patient Repositories”. In: *CoRR* abs/1608.00570 (2016). URL: http://arxiv.org/abs/1608.00570.

[41] Mesut Kaya, Derek Bridge, and Nava Tintarev. "Ensuring Fairness in Group Recommendations by Rank-Sensitive Balancing of Relevance". In: *RecSys*. 2020.

[42] Maurice G. Kendall. *Rank correlation methods*. London: Griffin, 1948. VII, 160. URL: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA%5C&SRT=YOP%5C&IKT=1016%5C&TRM=ppn+18489199X%5C&sourceid=fbw%5C_bibsonomy.

[43] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, and Young U. Ryu. "A group recommendation system for online communities". In: *International Journal of Information Management* (2010).

[44] Jong-Hun Kim, Daesung Lee, and Kyung-Yong Chung. "Item recommendation based on context-aware model for personalized u-healthcare service". In: *Multimedia Tools Appl.* 71.2 (2014), pp. 855–872. DOI: 10.1007/s11042-011-0920-0. URL: https://doi.org/10.1007/s11042-011-0920-0.

[45] Haridimos Kondylakis et al. "Development of interactive empowerment services in support of personalised medicine". In: *Ecancermedicalscience* 8 (2014), p. 400. ISSN: 1754-6605. DOI: 10.3332/ecancer.2014.400. URL: https://europepmc.org/articles/PMC3922652.

[46] Haridimos Kondylakis et al. "Patient Empowerment through Personal Medical Recommendations". In: *MEDINFO 2015: eHealth-enabled Health - Proceedings of the 15th World Congress on Health and Biomedical Informatics, São Paulo, Brazil, 19-23 August 2015*. 2015, p. 1117.

[47] Joseph A. Konstan et al. "GroupLens: Applying Collaborative Filtering to Usenet News". In: *Commun. ACM* 40.3 (1997), pp. 77–87. DOI: 10.1145/245108.245126. URL: http://doi.acm.org/10.1145/245108.245126.

[48] Caitlin Kuhlman and Elke Rundensteiner. "Rank aggregation algorithms for fair consensus". In: *Proceedings of the VLDB Endowment* 13.12 (2020).

[49] Defu Lian, Vincent W. Zheng, and Xing Xie. "Collaborative Filtering Meets Next Check-in Location Prediction". In: *WWW*. 2013.

[50] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. "Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts". In: *AAAI Conference on Artificial Intelligence*. 2016.

[51] Martin López-Nores, Yolanda Blanco-Fernández, José J Pazos-Arias, and Alberto Gil-Solla. "Property-based collaborative filtering for health-aware recommender systems". In: *Expert Systems with Applications* 39.8 (2012), pp. 7451–7457.

[52] Xin Luo, Yunni Xia, and Qingsheng Zhu. "Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization". In: *Knowledge-Based Systems* 27 (2012), pp. 271–280. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2011.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0950705111002073.

[53] Lucas Machado and Kostas Stefanidis. "Fair Team Recommendations for Multidisciplinary Projects". In: *2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019*. 2019, pp. 293–297.

[54] Judith Masthoff. "Group Recommender Systems: Aggregation, Satisfaction and Group Attributes". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci, Lior Rokach, and Bracha Shapira. Springer, 2015, pp. 743–776.

[55] Joseph F McCarthy and Theodore D Anagnost. "MusicFX: an arbiter of group preferences for computer supported collaborative workouts". In: *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. 1998, pp. 363–372.

[56] Omar Moling, Linas Baltrunas, and Francesco Ricci. "Optimal radio channel recommendations with explicit and implicit feedback". In: *RecSys*. 2012.

[57] Lihi Naamani Dery, Meir Kalech, Lior Rokach, and Bracha Shapira. "Iterative Voting Under Uncertainty for Group Recommender Systems". In: *RecSys*. 2010.

[58] Thuy Ngoc Nguyen, Francesco Ricci, Amra Delic, and Derek Bridge. "Conflict resolution in group decision making: insights from a simulation study". In: *User Modeling and User-Adapted Interaction* 29.5 (2019), pp. 895–941.

[59] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. "Fast Group Recommendations by Applying User Clustering". In: *ER*. 2012.

[60] Eirini Ntoutsi, Kostas Stefanidis, Katharina Rausch, and Hans-Peter Kriegel. "Strength Lies in Differences: Diversifying Friends for Recommendations through Subspace Clustering". In: *CIKM*. 2014.

[61] Mark O'connor, Dan Cosley, Joseph A Konstan, and John Riedl. "PolyLens: A recommender system for groups of users". In: *ECSCW*. 2001.

[62] Daniel Pfeifer. "Health Recommender Systems: Concepts, Requirements, Technical Basics and Challenges". In: *International Journal of Environmental Research and Public Health* 11.3 (2014), pp. 2580–2607.

[63] Evaggelia Pitoura, Georgia Koutrika, and Kostas Stefanidis. "Fairness in Rankings and Recommenders". In: *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*. 2020, pp. 651–654.

[64] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. "Fairness in Rankings and Recommendations: An Overview". In: *CoRR* abs/2104.05994 (2021). arXiv: 2104.05994. URL: https://arxiv.org/abs/2104.05994.

[65] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. "Fairness in Rankings and Recommendations: An Overview". In: *VLDB J.* 31.3 (2022), pp. 431–458.

[66] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. "Fairness in Rankings and Recommenders: Models, Methods and Research Directions". In: *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2021, pp. 2358–2361.

[67] D. Qin, X. Zhou, L. Chen, G. Huang, and Y. Zhang. "Dynamic Connection-based Social Group Recommendation". In: *IEEE TKDE* (2018).

[68] Jiarui Qin, Kan Ren, Yuchen Fang, Weinan Zhang, and Yong Yu. "Sequential Recommendation with Dual Side Neighbor-Based Collaborative Relation Modeling". In: *WSDM*. 2020.

[69] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. "Sequence-Aware Recommender Systems". In: *ACM Comput. Surv.* 51.4 (2018), 66:1–66:36. ISSN: 0360-0300.

[70] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. "Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks". In: *RecSys*. 2017.

[71] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews". In: *CSCW*. 1994.

[72] Tae Hyup Roh, Kyong Joo Oh, and Ingoo Han. "The collaborative filtering recommendation based on SOM cluster-indexing CBR". In: *Expert Systems with Applications* 25.3 (2003), pp. 413–423. ISSN: 0957-4174. DOI: https://doi.org/10.1016/S0957-4174(03)00067-8. URL: https://www.sciencedirect.com/science/article/pii/S0957417403000678.

[73] Dimitris Sacharidis. "Top-N Group Recommendations with Fairness". In: *SAC*. 2019.

[74] Amirali Salehi-Abari and Craig Boutilier. "Preference-Oriented Social Networks: Group Recommendation and Inference". In: *RecSys*. 2015.

[75] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. "Item-based collaborative filtering recommendation algorithms". In: *WWW*. 2001.

[76] Hanna Schäfer et al. "Towards Health (Aware) Recommender Systems". In: *Proceedings of the 2017 International Conference on Digital Health, London, United Kingdom, July 2-5, 2017*. 2017, pp. 157–161.

[77] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. "Fairness in Package-to-Group Recommendations". In: *WWW*. 2017.

[78] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. "An MDP-based recommender system." In: *Journal of Machine Learning Research* 6.9 (2005).

[79] Deepika Sharma, Gagangeet Singh Aujla, and Rohit Bajaj. "Evolution from ancient medication to human-centered Healthcare 4.0: A review on health care recommender systems". In: *International Journal of Communication Systems* (2019), e4058.

[80] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. "Explanation in case-based reasoning–perspectives and goals". In: *Artificial Intelligence Review* 24 (2005), pp. 109–143.

[81] Harald Steck. "Calibrated Recommendations". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2018. ISBN: 9781450359016. DOI: 10.1145/3240323.3240372. URL: https://doi.org/10.1145/3240323.3240372.

[82] Kostas Stefanidis, Eirini Ntoutsi, Mihalis Petropoulos, Kjetil Nørvåg, and Hans-Peter Kriegel. "A Framework for Modeling, Computing and Presenting Time-Aware Recommendations". In: *Trans. Large-Scale Data- and Knowledge-Centered Systems* 10 (2013), pp. 146–172.

[83] Julia Stoyanovich, Serge Abiteboul, and Gerome Miklau. *Data, Responsibly: Fairness, Neutrality and Transparency in Data Analysis*. International Conference on Extending Database Technology. Mar. 2016. URL: https://hal.inria.fr/hal-01290695.

[84] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. "FairGRecs: Fair Group Recommendations by Exploiting Personal Health Information". In: *Database and Expert Systems Applications - 29th International Conference, DEXA 2018, Regensburg, Germany, September 3-6, 2018, Proceedings, Part II*. 2018, pp. 147–155.

[85] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. "Multidimensional Group Recommendations in the Health Domain". In: *Algorithms* 13.3 (2020). ISSN: 1999-4893. DOI: 10.3390/a13030054. URL: https://www.mdpi.com/1999-4893/13/3/54.

[86] Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. "Fair Sequential Group Recommendations". In: *ACM SAC*. 2020.

[87] Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. "Fair sequential group recommendations". In: *SAC*. 2020.

[88] Maria Stratigi, Evaggelia Pitoura, Jyrki Nummenmaa, and Kostas Stefanidis. "Sequential group recommendations based on satisfaction and disagreement scores". In: *Journal of Intelligent Information Systems* (2021).

[89] Maria Stratigi, Evaggelia Pitoura, and Kostas Stefanidis. "SQUIRREL: A framework for sequential group recommendations through reinforcement learning". In: *Information Systems* 112 (2023), p. 102128. ISSN: 0306-4379. DOI:

https://doi.org/10.1016/j.is.2022.102128. URL: https://www.sciencedirect.com/science/article/pii/S0306437922001065.

[90] Maria Stratigi, Katerina Tzompanaki, and Kostas Stefanidis. "Why-Not Questions & Explanations for Collaborative Filtering". In: *Web Information Systems Engineering – WISE 2020*. Ed. by Zhisheng Huang, Wouter Beek, Hua Wang, Rui Zhou, and Yanchun Zhang. Cham: Springer International Publishing, 2020, pp. 301–315. ISBN: 978-3-030-62008-0.

[91] Xiaoyuan Su and Taghi M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques". In: *Advances in Artificial Intelligence* 2009 (2009).

[92] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. "Usage-Based Web Recommendations: A Reinforcement Learning Approach". In: *RecSys*. 2007.

[93] Nava Tintarev. "Explanations of recommendations". In: *RecSys*. 2007.

[94] Evangelos Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study*. Vol. 44. Jan. 2000. ISBN: 978-1-4419-4838-0. DOI: 10.1007/978-1-4757-3157-6.

[95] Lucas Vinh Tran et al. "Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation". In: *SIGIR*. 2019.

[96] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. *Fine-Grained Spoiler Detection from Large-Scale Review Corpora*. 2019. arXiv: 1905.13416 [cs.CL].

[97] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. "Explainable Recommendation via Multi-Task Learning in Opinionated Text Data". In: *ACM SIGIR*. 2018.

[98] Wen Wang et al. "Group-Aware Long- and Short-Term Graph Representation Learning for Sequential Group Recommendation". In: *SIGIR*. 2020.

[99] Lin Xiao et al. "Fairness-Aware Group Recommendation with Pareto-Efficiency". In: *RecSys*. 2017.

[100] Ronald R. Yager. "Fuzzy logic methods in recommender systems". In: *Fuzzy Sets and Systems* 136.2 (2003), pp. 133–149. ISSN: 0165-0114. DOI: https://doi.org/10.1016/S0165-0114(02)00223-3. URL: https://www.sciencedirect.com/science/article/pii/S0165011402002233.

[101] Sirui Yao and Bert Huang. "Beyond Parity: Fairness Objectives for Collaborative Filtering". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

[102] H. Yin et al. "Social Influence-Based Group Representation Learning for Group Recommendation". In: *ICDE*. 2019.

[103] Cong Yu, Laks V. S. Lakshmanan, and Sihem Amer-Yahia. "Recommendation Diversification Using Explanations". In: *ICDE*. 2009.

[104] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. "TV program recommendation for multiple viewers based on user profile merging". In: *User modeling and user-adapted interaction* 16.1 (2006), pp. 63–82.

[105] Quan Yuan, Gao Cong, and Chin-Yew Lin. "COM: A Generative Model for Group Recommendation". In: *KDD*. 2014.

[106] Zhang Yuyan, Su Xiayao, and Liu Yong. "A Novel Movie Recommendation System Based on Deep Reinforcement Learning with Prioritized Experience Replay". In: *ICCT*. 2019.

[107] Xiangyu Zhao et al. *Deep Reinforcement Learning for List-wise Recommendations*. 2019. arXiv: 1801.00209 [cs.LG].

[108] Guanjie Zheng et al. "DRN: A Deep Reinforcement Learning Framework for News Recommendation". In: *WWW*. 2018.

[109] Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee. "Fairness among New Items in Cold Start Recommender Systems". In: SIGIR '21. Virtual Event, Canada: Association for Computing Machinery, 2021. ISBN: 9781450380379. DOI: 10.1145/3404835.3462948. URL: https://doi.org/10.1145/3404835.3462948.

PUBLICATIONS

PUBLICATION
I

Fair sequential group recommendations

Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis

DOI: 10.1145/3341105.3375766. URL: https://doi.org/10.1145/3341105.3375766

Fair Sequential Group Recommendations

Maria Stratigi
Tampere University
Tampere, Finland
maria.stratigi@tuni.fi

Jyrki Nummenmaa
Tampere University
Tampere, Finland
jyrki.nummenmaa@tuni.fi

Evaggelia Pitoura
University of Ioannina
Ioannina, Greece
pitoura@cs.uoi.gr

Kostas Stefanidis
Tampere University
Tampere, Finland
konstantinos.stefanidis@tuni.fi

ABSTRACT

Recommender systems have been incorporated in our everyday life; from music to health recommendations, recommender systems have enhanced the users' experience. At the same time, with the expansion of social media, it is now easier than ever to form groups of people. As such, group recommenders have become more popular. Often, we consider the interaction between a group and the recommender system as a stand-alone process; the group requests some suggestions from the system and the system answers without any considerations to past interactions. A more realistic scenario is for a system to require access to history logs, and take them into account when recommending items for a group of users. Not only what items the system previously had recommended, but also, how well were these items received by the members of the group. In this work, we propose a sequential group recommender, which is aware of the past interactions of the group with the system. We introduce the notion of *satisfaction* that describes how relevant are the recommended items to each member of the group. We utilize satisfaction in a novel aggregation method that achieves to make our model fair for all members of the group. We show with experimental results that the typical group recommendation approaches are substandard to our proposed method.

KEYWORDS

Recommender systems, sequential recommendations, group recommendations, fair recommendations.

1 INTRODUCTION

Recommendations have been integrated into many of the services available to users in recent times. From listening to music to health information, recommender systems are employed to make the user experience better and smoother. In most cases, recommender systems provide users with a list of data items that are most relevant to them. Two main methods appear to produce such lists of items. The content-based method [24] and the collaborative filtering method [29]. In the content-based case, the system recommends to the user items that are similar to other items that the user has already consumed in the past. This requires previous knowledge about the items which is often hard to obtain. In contrast, collaborative filtering (CF) requires data regarding user preferences for specific items. The main idea behind a CF recommender system is the following: given a target user, the system finds similar enough users to him/her, often called *peers*. Then the items that the peers have shown a preference for (mostly in the form of ratings, but often other forms of feedback are used, such as textual review and binary format of like/dislike) are examined and used as input into a relevance function, which produces the list of relevant items to the target user. Collaborative filtering is a very powerful tool that enables recommender systems to provide far more accurate and specialized recommendations [3].

With the expansion of social media, another form of recommendations has emerged; namely the group recommendations [1, 21, 22]. Instead of a single user requesting recommendations from the system, a group can make a query as well. A standard example of group recommendations is the following: a group of friends wants to see a movie. Each friend has his/her own likes and dislikes. The system needs to properly balance them, and offer to the group a list of items that has a degree of relevance to each member. Recently, there is some research on the selection process of the items. One approach is to create a pseudo user by combing the data of each group member, and then apply a standard recommendation method. The second and most used approach is to apply a recommendation method to each member individually, and then aggregate the separate lists into one for the group. The aggregation phase of the approach is the object of much research done on group recommendations.

There are many different criteria one can take into account during the aggregation stage. One such criterion is *fairness* [19, 30–32]. A basic definition of fairness is to recommend the item that has the best relevance to the group. Instinctively, one way to produce such an item is to calculate the average score across all the group members' preference scores for that item. In such a way, all members of the group are considered equals. However, this has a big drawback. Consider, for instance, the following scenario with a group of three users. Two of them are quite similar to each other, while the third is not. By using the average method the opinion of the last user is lost. An alternative approach is to use the minimum function rather than the average. This way, the user with the minimum preference score will act as a veto to the rest of the group. The least misery approach, as it is called, has a drawback as well. It only takes the opinion of one group member under consideration, while ignoring the others. Subsequently, the system in most cases, recommends an item that is acceptable for all members, but it will be an item with a somehow *low* preference score. The need to combine the equality of the average method and the inclusivity of the least misery method is the driving force behind our work in this paper.

The previous approaches for computing recommendations have a hidden element to them. We imply that each time a group is using the system is distinct from the previous ones and that the system has only one state without keeping a history log. But this is not a realistic scenario. A user, or in our case a group of users, interacts with the system multiple times. So, the system should recommend different items at each iteration, while retaining knowledge from past interactions, so as to keep the output diverse. That is, we do not want to always recommend the same item or the same set of items. We want a system that has a memory and can adjust its recommendations accordingly. Continuing with a group movie example, where the same group of friends requests a movie recommendation each week, both the average and least misery approaches fail through all iterations of the system. In the case of average, the outlier user is never satisfied, while in the case of least misery, the system recommends movies that are not highly relevant to anyone in the group.

In this work, we address the problem of unfairness that is generated by these methods when applied to sequential group recommendations. A simple working scenario is as follows: we have a group of friends that meets in regular times to watch a movie. We want to recommend each time a new movie for the group. Figure 1 gives an experimental example of such a scenario. We take a group of 5 members that ask the system for recommendations 5 different times. Each time the system reports to the group 10 items. In Figure 1-left, we have formed the group list, by using the average aggregation method, while in Figure 1-right, the least misery one. In this example, we count the degree of *satisfaction* for each member, which is calculated by measuring how relevant are the group list's items, over the best items for each member[1]. In both scenarios, User 4 has a very low satisfaction score (for least misery is always 0), which implies that almost none of the reported items are of interest to him/her. It is evident that the recommender system is unfair to him/her, and that unfairness continues throughout the iterations. Ideally, each new group recommendation should take into account what has happened in the past. Not only what movies have already been recommended, but at which degree each member of the group was satisfied with that recommendation.

The notion of multiple iterations of recommendations allows us to make a conjecture. If a user is not satisfied with a particular round of recommendations, then he/she was either satisfied in a previous or will be satisfied in the next round. We introduce the notion of *satisfaction* to estimate the degree of satisfaction in the recommendations for each group member, after each iteration of the system. We apply this satisfaction score during the aggregation phase of the group recommendation method as a weighting function on the individual preference scores of the group members. This way, the users that were not satisfied in the previous round will have more weight in the next iteration. Additionally, a member is not continuously biased against (as may be the case of simple average), since the calculation of the satisfaction scores is done dynamically at each iteration. Finally, we take into account the opinions of the entire group (something that least misery was lacking).

The contributions of our work are the following:

- We introduce the notion of sequential group recommendations. We propose that when adding the dimension of multiple iterations to typical group recommendation approaches, such as the average and least misery aggregation methods, the results do not ensure user satisfaction for all group members.
- We propose the concept of *satisfaction*. Each member of the group has a degree of satisfaction for the items recommended at each iteration, as well as an overall satisfaction, gained by all the previous iterations of the group.
- We propose a sequential group recommendation model that takes into account the previous interactions of the group with the system and alters the influence that a group member has on the formation of the group recommendation list.
- We experimentally show that our proposed model is superior to the standard group recommendation approaches.

The rest of the paper is structured as follows: Section 2 describes the related work. Section 3 presents basic concepts on recommender systems, and Section 4 introduces our approach for sequential group recommendations, targeting at ensuring fair results for all group members. Section 5 presents our experimental setup, and Section 6 presents our evaluation results. Finally, Section 7 concludes the paper with a summary of our contributions and directions for future work.

2 RELATED WORK

A recommender system aims to provide to a user, items that are relevant to him/her, by exploiting already available user information – profile, preferences, etc. One of the most used approaches for producing recommendations is the collaborative filtering approach. With more than a decade of research in the area, there are many different solutions to the problem [3]. Most of them can be divided into two main categories: memory-based and model-based algorithms. Memory-based algorithms [15, 28], employ a user-ratings matrix that contains the ratings each user has given to items. They utilize this matrix to find similar users to a target user, by applying a similarity function. The final prediction is made by examining the ratings of similar users, or as they often called *neighbors or peers*. Model-based algorithms [7], first construct a model to represent the behavior of the users and, therefore, to predict their ratings. In this work, we utilize memory-based collaborative filtering.

2.1 Group Recommendations

Group recommendation is another field with a significant research background. There are two main approaches to group recommendation: virtual user and recommendation aggregation [13]. In the former approach, we combine the profiles and ratings of each group member to form a virtual user so that a standard recommendation approach can be applied. In the latter approach, we apply a standard recommendation algorithm to each group member individually and aggregate their lists into one. In this work, we follow the latter approach, since it is more flexible [23] and offers opportunities for improvements in terms of effectiveness.

During the aggregation phase of this approach, many criteria can be taken into account. [36] proposes a group recommendation model that takes into account the influence that each member has

[1]For more details see Section 4 – Equation 1.

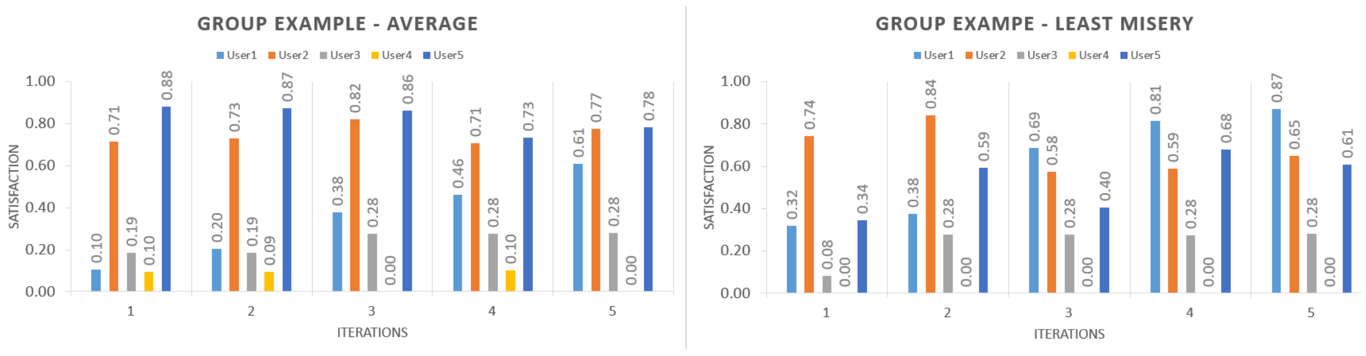


Figure 1: Example for a group with 5 members, and a group recommendation list with 10 items. We consider 5 iterations. We utilize the average (left) and least misery (right) aggregation methods.

on the final choice for the group. They state that a member has more influence on the group if he/she is more knowledgeable about the items that are recommended. In our work, we propose that, if there is a group member that is more dissatisfied than the rest, then that member will have more influence in the group decision. [4] learns the aggregation strategy from data, which is based on the recent developments of attention network and neural collaborative filtering (NCF), while we dynamically change our proposed aggregation method based on the satisfaction of the group members. While we focus on a relatively small group of friends (5 in our experiments), [25] offers a novel approach to producing recommendations for a large group of people, by dividing the big group into different interest subgroups. For each subgroup, they find a potential candidate set of media-user pairs and finally aggregate the CF produced recommendation lists for each pair. [16] proposes a two-phase group recommender that, similar to our work, tries to satisfy all the group members. In the first phase, they try to satisfy the whole group, while in the second they try to satisfy the members individually, by filtering out items that are irrelevant to each member. We incorporate these two phases into our proposed aggregation method.

2.2 Fairness in Group Recommendations

There are many different approaches to achieving fairness in group recommendations. One approach is to calculate fairness based on the game and voting theory. [5] solves conflicts of interest between members of heterogeneous random groups by utilizing the Non-Cooperative Game Theory [34]. [20] assumes that the recommender system has probabilistic knowledge about the distribution of users' ratings, and utilizing the voting theory recommends to the group a "winning" item. Finally, [9] proposes a new group recommendation method by allowing a group member to commend on the choices of the rest of the group. This allows each user to get new recommendations similar to the proposals made by other group members and to communicate the rationale behind their own counterproposals.

Another approach to achieving fairness in group recommendation is presented in [1]. It introduces the *consensus function* that similarly to our work takes into account the opinion of the group as it is given by the average method and the disagreement between users. They define the disagreement of users as either the average of pair-wise relevance differences for the item among group members or a variance disagreement function that computes the mathematical variance of the relevance scores for the item among group members. [30] proposes two definition of fairness: *fairness proportionality* and *envy-freeness*. In the former, the user u considers the list of recommended items fair for him/her, if there are at least m items that the user likes. In the latter, u considers the package fair, if there are at least m items for which the user does not feel envious. [35] presents yet another definition of fairness. They define a utility score for each group member based on the relevance that the recommended items have on them. They model fairness as a proximity of how balanced the utilities of users are when group recommendations are given.

All of the works mentioned above in achieving fairness only consider one instance of group recommendations and do not take into account the sequential group recommendation problem, as we do in this work.

2.3 Sequential Recommendations

Sequential recommendations is a relatively new area of research. In general, there are three categories of sequential recommenders, and they are divided based on how many past user interactions they consider: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* [26]. In the first approach, only the last N user actions are considered [6, 17, 18]. This is because the system has logged a huge amount of historical data for the user, with many of them be duplicates, which do not offer relevant information to the system. Last-N interactions-based recommendations are typically location-aware recommendations. In session-based recommendations, only the last interaction of the user with the system is used. They are typically found in news [8] and advertisement [12] recommender systems. In the last category of sequential recommenders – session-aware recommendations, we have information about both the last interaction of the user with the system, as well as the history of the user. These recommenders are often implemented in e-commerce or for app suggestions [10, 14, 27]. In our work, we use the last method to approach sequential group recommendations. The above classification has been done for single user recommenders, and to our knowledge, our work is the first one in sequential group recommendations, that handles the notion of fairness.

3 BASIC CONCEPTS

In this section, we introduce a simple group recommendation process. This is a stand-alone process and does not take into account any past interactions that the group may have had with the system. The group recommendation approach we use, is the aggregation of the individual group members' preference lists into one group preference list. We define a preference list to be the list we recommend either to each member individually or to the group.

Specifically, let U be a set of users and I be a set of items. Each user $u_i \in U$ has given a rating from 1 to 5 to an item $d_z \in I$ represented as $r(u_i, d_z)$ The subset of users that rated an item $d_z \in I$ is denoted by $U(d_z)$, while the subset of items rated by a user $u_i \in U$ is denoted by $I(u_i)$. Let's assume also a fixed group of users G of size n, $G \subseteq U$. Having applied a single user recommendation algorithm to all group members in G and produced their preference lists $A_{u_1}, \dots, A_{u_n}$, the next step is to aggregate these lists into one. After the aggregation phase, each item will have just one group preference score and the top k will be reported back to the group as the final recommendations.

Typically, two well established aggregation methods are used [2]. The first aggregation approach we examine is the *average* method. The main idea behind this approach is that all members are considered equals. So, the group preference of an item will be given be averaging its scores across all group members:

$$avgG(d_z, G) = \frac{sum_{u_i \in G} p(u_i, d_z)}{|G|},$$

where $p(u_i, d_z)$ gives us the preference score of d_z for user u_i (computed by a standard single user recommendation algorithm). The second aggregation method is *least misery*, where one member can act as a veto for the rest of the group. In this case, the group preference score of an item d_z is the minimum score assigned to that item in all group members preference lists:

$$minG(d_z, G) = min_{u_i \in G} p(u_i, d_z).$$

4 SEQUENTIAL GROUP RECOMMENDATIONS

In this section, we introduce the notion of multiple rounds to the previous group recommendation process. Consequently, we do not consider each group query to the system as a stand alone process, but as a sequence of queries submitted to the system by the same group. We call each group query to the system an *iteration*, and each iteration has the main components of a standard group recommendation method: single user recommendations followed by the aggregation phase.

Formally, let $\mathcal{GR}$ be a sequence of μ group recommendations $(Gr_1, \dots, Gr_\mu)$. We use $p_j(u_i, d_z)$ for the preference score of user u_i for item d_z at iteration j, $1 \leq j \leq \mu$. These scores are estimated by a single user recommendation algorithm. After the aggregation phase of the group recommender system, we use $gp_j(G, d_z)$ to denote the preference score of item d_z for the group G as a whole, as estimated by the group recommender at iteration j.

This new model, in contrast to the previous one, introduces the notion of *multiple iterations* or *sequence* of recommendations. We use these interactions, to alter the output of the recommender system, in such a way that if a user was not satisfied in a previous iteration, potentially, he/she will be satisfied during the current one.

4.1 Satisfaction Measure

To examine the effectiveness of our group recommender algorithm through a series of iterations, we need to define a measure that will help us elevate the problem from examining each iteration independently, to examining a series of iterations. We introduce the notion of *satisfaction* that represents the gratification of each group member for the recommended items after each iteration of the system. We define two variations of satisfaction: *single user satisfaction* and *group satisfaction.*

4.1.1 Single User Satisfaction. First, we want to provide a formal measure of the degree of the satisfaction of each user u_i in G to the group recommendation Gr_j received at step j. We do so, by comparing the quality of recommendations that the user receives as a member of the group with the quality of the recommendations that the user would have received as an individual.

Assume that Gr_j involves items $i_{j,1}, i_{j,2}, \dots i_{j,k}$. Furthermore, let $A_{u_i,j}$ be the list with the top-k items $a_{j,1}, a_{j,2}, \dots a_{j,k}$ for user u_i, that is, the k items with the highest prediction scores for user u_i. Our goal is to directly compare the user's satisfaction from the group recommendation list with the ideal case for that user. This gives us a more clear view of the satisfaction of the user than the average method since we take into account the top items for each user, and not only the top items for the group. Formally:

$$sat(u_i, Gr_j) = \frac{GroupListSat(u_i, Gr_j)}{UserListSat(u_i, A_{u_i,j})} \quad (1)$$

$$GroupListSat(u_i, Gr_j) = \sum_{d_z \in Gr_j} p_j(u_i, d_z) \quad (2)$$

$$UserListSat(u_i, A_{u_i,j}) = \sum_{d_z \in A_{u_i,j}} p_j(u_i, d_z) \quad (3)$$

With the function *GroupListSat* (Equation 2), we calculate the user's satisfaction based on the group recommendation list. For every item in Gr_j, we sum the score as they appear in each user's $A_{u_i,j}$. The function *UserListSat* (Equation 3), calculates the ideal case for the user, by simply sum the scores of the k top items in the user's $A_{u_i,j}$. This way, we are able to normalize the user's satisfaction score. For example, if a user gives mainly low scores to items, then Equation 1 is able to compensate for it.

Note that in both Equations 2 and 3, we do not use the scores as they appear in the group list, but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process, rather distorts the individual opinions of the group members, we opt to take into consideration only the personal preference scores of each group member.

Still, Equation 1 remains *static*, in the sense that we calculate the satisfaction of a user for one iteration only. As stated before, what we want is to define a measure that takes into consideration the satisfaction scores from previous iterations of the system. Such a score will represent the overall satisfaction of each group member with the entirety of the μ group recommendations.

Definition 4.1 (Overall Satisfaction). The overall satisfaction of user u_i with respect to a sequence $\mathcal{GR}$ of μ iterations is the average of the satisfaction scores after each iteration:

$$satO(u_i, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u_i, Gr_j)}{\mu} \tag{4}$$

4.1.2 Group Satisfaction. Having defined the satisfaction score of each group member we can now define the satisfaction score of the entire group. Specifically, the satisfaction of the group G with respect to a group recommendation list Gr_j is defined as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j) = \frac{\sum_{u_i \in G} sat(u_i, Gr_j)}{|G|} \tag{5}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$, as:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u_i \in G} satO(u_i, \mathcal{GR})}{|G|} \tag{6}$$

This measure indicates if the items we report to the group, are acceptable to its members. Higher group satisfaction means that the group members are satisfied with the recommendations. However, since we average the members satisfaction scores a dissatisfaction of a user can probably be lost in the computations. To counter this problem, we focus as well on the potential disagreements between the users in the group. For representing such *disagreement*, we define the *groupDis* measure:

$$groupDis(G, \mathcal{GR}) = \\ max_{u_i \in G} satO(u_i, \mathcal{GR}) - min_{u_i \in G} satO(u_i, \mathcal{GR}) \tag{7}$$

Intuitively, we define the disagreement of the group, to be the difference in the overall satisfaction scores between the most satisfied and the least satisfied member in the group. Ideally, we want this measure to take low values, as that will indicate that the group members are all satisfied to the same degree. Higher *groupDis* values will demonstrate that at least one member of the group is biased against.

4.2 Problem Definition

Our sequential group recommender system needs to achieve two independent objectives that are nonetheless critical to the success of our model. The first objective considers the group as an entity: we want to offer to the group the best possible results. The second objective considers the group members independently: we want to behave as fairly as possible towards all members.

Definition 4.2 (Fair Sequential Group Recommendation). Given a group G, the sequential group recommender produces at the μ iteration a list of k items Gr_μ, $Gr_\mu \in \mathcal{GR}$ that:

(1) Maximizes the overall group satisfaction, $groupSatO(G, \mathcal{GR})$, and
(2) Minimizes the variance between users satisfaction scores: $groupDis(G, \mathcal{GR})$.

To achieve the first objective, we target at maximizing *groupSatO*. This means that we require the items with the highest preference scores for the group as a whole. Since *groupSatO* expresses the average satisfaction of the group members, and the dissatisfaction of just one member is easily lost, we also need to achieve the second objective, to minimize *groupDis*, which is the representation of the degree of dissatisfaction between the members of the group.

Depending on the similarity between the group members, these two objectives may be conflicting with each other. A simple example, is a group of three members, two that are highly similar to each other, and one that is very dissimilar to them. To achieve high *groupSatO* values, we need to recommend items that are relevant to the two similar users, so as to increase the average satisfaction of the group. On the other hand, by doing so, *groupDis* will take high values as well since we do not address the needs of the third member. Overall, we need to recommend items that are not just good enough for all members – since that still returns low *groupSatO* values, but items that have high relevance to the group, without sacrificing the opinions of the minority.

4.3 Sequential Hybrid Aggregation Method

As explained above, both the *average* and the *least misery* aggregation methods have drawbacks, when we consider them for sequential recommendations. At the same time, both have advantages (namely, equality for average, and inclusion of all opinions for least misery) that are an asset for a recommender system. Furthermore, it has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem [35]. That is something we want to achieve throughout the multiple iterations of the system, to be equally fair to all members of the group. To circumvent the problem and to capitalize on the advantages of the average and the least misery aggregation methods, we propose a novel aggregation method that is a weighted combination of them. We call the method *sequential hybrid aggregation method.*

$$score(G, d_z, j) = \\ (1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j) \tag{8}$$

The function $avgScore(G, d_z, j)$ returns the score of the item d_z as it is computed by the average aggregation method during iteration j, and function $leastScore(G, d_z, j)$ returns the least satisfied user's score of d_z at iteration j. The variable α takes values from 0 to 1. If $\alpha = 0$, then the sequential hybrid aggregation method becomes average aggregation, while when $\alpha = 1$, it transforms to a modified least misery aggregation, where we only take into account the preferences of the least satisfied member.

Intuitively, when $\alpha = 0$, our method satisfies the first part of the fair sequential group recommendation problem (Definition 4.2) since the average aggregation considers the best options for the group as a whole. The benefits of $\alpha = 1$ can be seen for higher values of μ, since at the first iterations the group disagreement may remain high since it considers the overall satisfaction of the users (Equation 4).

Given that our goal is to fulfill both objectives of the problem definition, we need to set the value of α between 0 and 1. Furthermore, we want this value to self-regulate, to more effectively describe the

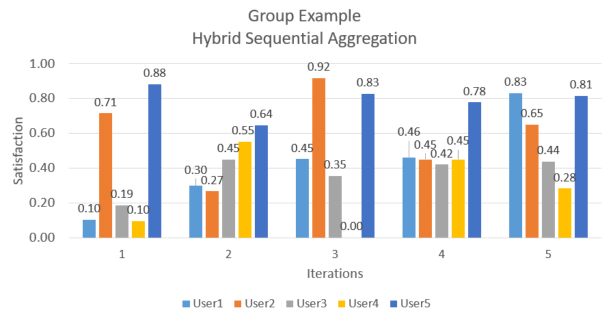


Figure 2: Example for the same group as in Figure 1. We consider 5 rounds. We utilize the Sequential Hybrid Aggregation method.

consensus of the group. Thus, we set the value of α dynamically in each iteration by subtracting the minimum satisfaction score of the group members in the previous iteration, from the maximum score.

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}) - min_{u \in G} sat(u, Gr_{j-1}) \qquad (9)$$

When $j = 1$ (the first iteration of the system), then $\alpha = 0$, and the aggregation method reverts to that of a classic average aggregation.

By having this dynamically calculated α, we counteract the individual drawbacks of average and least misery. Intuitively, if the group members are equally satisfied at the last round, then α takes low values, and the aggregation will closely follow that of an average, where everyone is treated as an equal. On the other hand, if one group member is extremely unsatisfied in a specific iteration, then α takes a high value and promotes that member's preferences on the next iteration. Formally:

Proposition 1. *Let u be a user in a group G, such that, $sat(u, Gr_{j-1}) < sat(u_i, Gr_{j-1})$, $\forall u_i \in G \backslash \{u\}$. Then, $\exists u_y \in G \backslash \{u\}$, such that, $sat(u_y, Gr_j) < sat(u, Gr_j)$.*

Proof. For the purpose of contradiction, assume that $sat(u, Gr_j) < sat(u_y, Gr_j)$, $\forall u_y \in G \backslash \{u\}$. Then, this means that u is the least satisfied user at iteration j, which in turn means that α takes values not leading towards a least misery approach at this iteration, meaning that u was not the least satisfied user at iteration $j - 1$, which is a contradiction. □

To exemplify this, we run the same experiment as in Figure 1, for the same group, but now we utilize the sequential hybrid aggregation method. The results appear in Figure 2. Here, we can observe that a group member that was not satisfied in the previous iteration of the system, is satisfied in the next. A good example of this is User 4 where in the first iteration has a very low satisfaction score, and in the second has a higher one. This is a clear improvement over the results shown in Figure 1, where User 4 was always the least satisfied member of the group.

4.4 Algorithm

Our sequential hybrid group aggregation method is described in Algorithm 1. The input to the algorithm is the group G, the iteration j, and the size k of the group recommendation list Gr_j, which we report to the group after each iteration is finished. In Line 1, we apply a single user recommendation algorithm to each group member and save their preference lists into the variable A. In Line 2, we define a set that contains all the items that appear in the members' preference lists Gl and, in Lines 3–5, we populate the set. In Lines 6–10, we calculate the α; if it is the first iteration meaning $j = 1$ (Line 7), then $\alpha = 0$, otherwise, we use Equation 9 (Line 9) to calculate it. In Lines 11–24, we perform the sequential hybrid aggregation method. For all the items in Gl, we calculate the item score using Equation 8 (Line 12) and insert them in the group list Gr_j (Line 13). After we have examined all the items in Gl, we sort the items in the group list (Line 15), and finally, we report the top-k of them to the group (Line 16). The complexity of our algorithm is $O(n \log n)$ due to the time complexity of the sorting function, where $n = |Gl|$.

```
Data: Group G, $j$ top $k$
Result: Group Recommendation List: $Gr_j$
1  $A \longleftarrow RS(G)$;
2  $Gl \longleftarrow \emptyset$;
3  for $u_i \in G$ do
4      $Gl \longleftarrow Gl \cup A_{u_i}$;
5  end
6  if $j$=1 then
7      $\alpha_j = 0$;
8  else
9      $\alpha_j = maxSat(G, j-1) - minSat(G, j-1)$;
10 end
11 for item $d_z$ in $Gl$ do
12     $score(G, d_z, j) =$
         $(1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j)$;
13     $Gr_j \longleftarrow Gr_j \cup score(G, d_z, j)$;
14 end
15 $sort(Gr_j)$;
16 $report(top(Gr_j, k))$;
```
Algorithm 1: Sequential Group Recommendation Algorithm

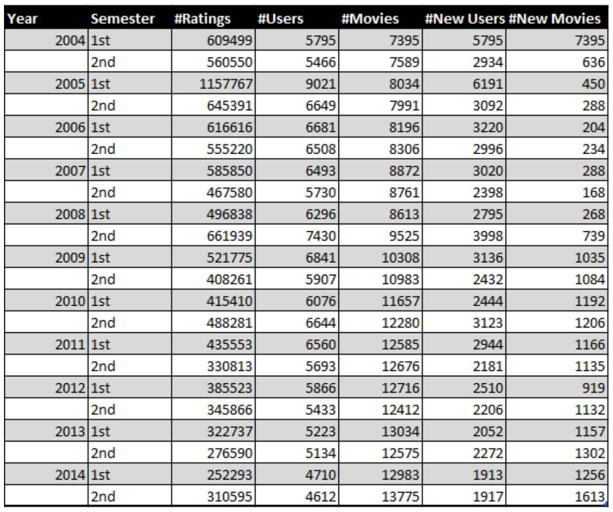
5 EXPERIMENTAL SETUP

In this section, we describe the two pre-processing tasks needed for the experimental evaluation of our sequential group recommender system: the dataset splitting and the group formation.

5.1 Dataset

For the experimental evaluation of our sequential group recommender system, we use the 20M MovieLens Dataset [11]. It contains 20.000.263 ratings across 27.278 movies. These data were created by 138.493 users between January 09, 1995, and March 31, 2015. The dataset was generated on March 31, 2015. To simulate multiple iterations of suggestions, we split the dataset into chunks. Each chunk is added to the system, representing new information, and along with the already existing data in the system, is used for locating the suggestions for the next iteration.

Initially, we divide the dataset into two parts of roughly the same size. The first part that contains the ratings given between January 1994 and December 2003, is the starting dataset of our recommender.



Year	Semester	#Ratings	#Users	#Movies	#New Users	#New Movies
2004	1st	609499	5795	7395	5795	7395
	2nd	560550	5466	7589	2934	636
2005	1st	1157767	9021	8034	6191	450
	2nd	645391	6649	7991	3092	288
2006	1st	616616	6681	8196	3220	204
	2nd	555220	6508	8306	2996	234
2007	1st	585850	6493	8872	3020	288
	2nd	467580	5730	8761	2398	168
2008	1st	496838	6296	8613	2795	268
	2nd	661939	7430	9525	3998	739
2009	1st	521775	6841	10308	3136	1035
	2nd	408261	5907	10983	2432	1084
2010	1st	415410	6076	11657	2444	1192
	2nd	488281	6644	12280	3123	1206
2011	1st	435553	6560	12585	2944	1166
	2nd	330813	5693	12676	2181	1135
2012	1st	385523	5866	12716	2510	919
	2nd	345866	5433	12412	2206	1132
2013	1st	322737	5223	13034	2052	1157
	2nd	276590	5134	12575	2272	1302
2014	1st	252293	4710	12983	1913	1256
	2nd	310595	4612	13775	1917	1613

Figure 3: Divided by semester: Ratings, unique users and movies, as well as the number of new users and movies that appear in each chunk

This gives us an initial dataset that consists of 8.381.255 ratings, 73.519 users and 6.382 movies. We initiate the system with this starting dataset to avoid any issues emanating from the cold start problem[2]. The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of six months. During the first iteration, the system will have access only to the initial dataset. When that iteration ends, the system will be enhanced with one additional chunk (first semester of 2004), and after each subsequent iteration, the system will be given access to the next chunk (second semester of 2004, fist semester of 2005, etc.).

Figure 3 shows the detailed information on the contents of each chunk. With the last two columns, we report the number of users and movies respectively that appear for the first time during the specific semester. As we can see, the ratings are not evenly split between the semesters. There is a higher number of ratings in the years 2004 to 2010, in some cases even triple the amount. As it is expected, the number of movies that appear in each chunk increases as the time passes, since the users can rate old movies as well. At the same time, the number of users is relatively the same across all chunks. The disinterested of the users to give ratings as time passes is not a complication in our evaluations since the total number of available ratings is enough for our needs.

5.2 Group Formation

We will evaluate our sequential group recommender on stable groups. This means that the members of the group do not change between iterations, and all the members are present for each recommendation. We will examine our recommender on four types of groups, based on the similarity shared between the members. We target at covering groups with different semantics, starting from groups with similar users to groups with dissimilar ones. We calculate the similarity between the members of the group using only the starting dataset and not the entirety of it since we want the relationship between the users to be present from the first iteration.

The similarity between the group members is calculated using the Pearson Correlation similarity measure [28], which takes values from −1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity.

$$s(u_i, u_l) = \frac{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})(r(u_l, d_z) - \bar{r}_{u_l})}{\sqrt{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})^2} \sqrt{\sum\limits_{d_z \in X} (r(u_l, d_z) - \bar{r}_{u_l})^2}} \tag{10}$$

where $X = I(u_i) \cap I(u_l)$ and $\bar{r}_{u_i}$ is the mean of the ratings in $I(u_i)$, i.e., the mean of the ratings of user u_i.

We consider two users to be highly similar to each other, if they share above 0.5 similarity score, while dissimilar when they have -0.5 or lower similarity score. The types of groups, we are considering are the following:

- **4 similar – 1 dissimilar:** The four members of the group share a high similarity score, while the dissimilar one shares a low similarity score with the rest of the group members.
- **3 similar – 2 similar:** We divide the group into two subgroups. The members of each subgroup are similar to each other, while at the same time, the subgroups are dissimilar to one another, i.e., all members of one subgroup are dissimilar to all members of the other subgroup.
- **3 similar – 1 dissimilar – 1 dissimilar:** We divide the group into three separate subgroups: one that contains three members and two subgroups that each contain just one user. All subgroups are dissimilar to each other, while the three members belonging in the same subgroup are similar to each other.
- **5 dissimilar:** All members of the group are dissimilar with each other.

For each group type, we generate 100 groups, and each user can only participate in one group per category. We consider only groups of five members since that is a realistic scenario.

6 EVALUATION

In this section, we go over our experimental procedure and the variables we took into consideration during the evaluation of the sequential group recommender. We present the results of this evaluation that showcase the advantages of our sequential hybrid aggregation method.

6.1 Experimental Procedure

In our experiments, we will examine the behavior of four types of groups, considering 100 different groups per type. For each group in a set, we perform sequential group recommendations for various values of μ, as described in Algorithm 1. For all the groups in the set, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDis(G, \mathcal{GR})$. Additionally, we utilize an F-score measure (Equation 11), namely the harmonic mean of the $groupSatO$ and $groupDis$ measures, that provides a good indication of the users'

[2]The cold start problem in collaborative filtering appears when there is not enough information about a user and we are unable to find similar other users to him/her. This problem is beyond the scope of this research, and to overcome it, we initialize our system with a large enough dataset.

satisfaction and the agreements between the users in the group. Taking into account the input functions that F-score needs, we use $1 - groupDis$ to simulate the group agreement.

$$F\text{-}score = 2\frac{groupSatO * (1 - groupDis)}{groupSatO + (1 - groupDis)} \tag{11}$$

We find similarities between users, by utilizing the Pearson Correlation. We consider two users as similar if the similarity is greater than 0.7, and if they have rated more than five identical items.

To predict preference scores for a user, we use the Weighted Sum of Others Ratings [33], and we take into account only the top 100 most similar users to him/her. We recommend to the group the 10 items with the highest group preference score. In our case, we assume that the system does not recommend items to the group that it has previously recommended.

$$p(u_i, d_z) = \bar{r}_{u_i} + \frac{\sum_{u_l \in (P_{u_i} \cap U(d_z))} s(u_i, u_l)(r(u_l, d_z) - \bar{r}_{u_l})}{\sum_{u_l \in (P_{u_i} \cap U(d_z))} |s(u_i, u_l)|} \tag{12}$$

6.2 α Experiments

To examine the behavior of the sequential hybrid aggregation method, we first examine the values that the variable α (Equation 9) takes during the iterations of the system. We examine the performance of α through 15 iterations. In Figure 4, we report the average α values of the 100 groups per type, per iteration. At the first iteration, the α takes the default value 0. During the next iterations, the α values increase in a close to linear form. This is expected since by design the groups have at least one outlier member that is dissimilar to the rest of the group. This guarantees that at least one member is not satisfied during an iteration of the system. The high values of α in the later iterations are also the result of the dataset splitting. At the later iterations, the majority of the dataset has already been given as input to the system, and statistically, the best items for the group as well as for the individual members have already been recommended. Since per our scenario our system cannot suggest items that have already been recommended, the remaining items are less preferable to the group as a whole and each member individually.

An additional observation is the different behaviors of the four group types. As already stated, the groups are formed in such a way, as to always have at least one member being a minority. The α values offer a first impression of the implications these types of groups have on sequential recommendations. Remember that α is different than the group disagreement (Equation 7) since it only takes into account the members' satisfaction for the previous iteration of the system. During the first iterations of the system, there is a little variation in the values of α. The clear distinction becomes obvious in the later iterations, where we can observe that the more diverse the group becomes, the more higher the values of α get. This is expected since the more distinct opinions in the group are, the higher the disagreement between the user becomes, and thus the values of α become higher.

6.3 Group Type Experiments

To examine the effectiveness of the sequential group recommender, we consider the group satisfaction, *groupSatO* (Equation 6) and the disagreement between the users, *groupDis* (Equation 7), after a number of iterations μ. For each type of group, we perform the experiments for different number of iterations, $\mu \in [5, 10, 15]$. Figures 5, 6, 7 and 8, show the average of the group satisfaction and group disagreement scores for the 100 groups per group type, respectively.

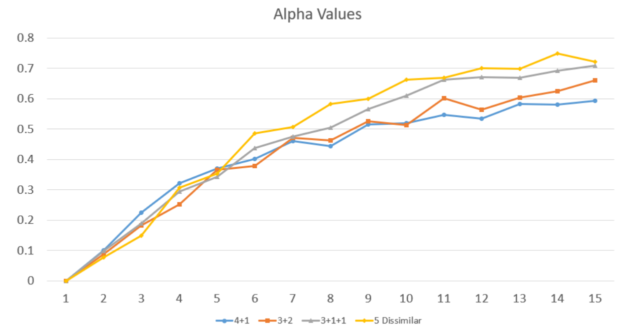


Figure 4: Average α values over 100 groups per group type, at each iteration.

We perform the tests for different values of α, $\alpha \in [0, 0.4, 1, \textit{dynamic}\ \alpha]$, as well as the pure Least Misery approach (LM). For $\alpha = 0$, we have a standard average aggregation method. For $\alpha = 1$, we have a slightly altered least misery aggregation method, where instead of assigning the lowest score among the individual group members' preference scores to the item as a group preference score, we assign to the item the score as it appears on the preference list of the least satisfied user. We also consider a static value for α, which is 0.4, selected as the one with the best results, after extensive experimentation on different α values.

When considering the overall group satisfaction, we observe that for all group types it decreases by the same degree. This is because in the later iterations of the system the best items for the group have already been reported, and as stated we do not recommend items that have already been recommended in previous iterations. When comparing the group satisfaction for the different values of α, we observe that the average method slightly outperforms the dynamic α. For 15 iterations, the average method offers a better overall group satisfaction by a slight factor, since it tries to offer the best results to the group as a whole. The dynamic α, also clearly outperform the LM method, especially the more diverse the groups become. By far the worst performance is for $\alpha = 1$, since in each iteration of the system, we consider only one user, without taking into account the opinions of the rest of the group.

Regarding the group disagreement, LM has the worst results followed by the average method ($\alpha = 0$), while the dynamic α has the best. This is because, the average method ignores the minority opinion in the group, which is reflected in the high group disagreement values. LM does not work well for diverse groups since the preferences of the members greatly differ from each other. This becomes more apparent the more dissimilar the group members are, as shown in Figure 8. The benefits and drawbacks of $\alpha = 1$ are apparent in our evaluations. We can observe high disagreement in the first iterations of the system, but later, we have better results. This is more clear in more diverse groups, such as the ones in Figure 8, where during the later iterations it achieves the second best disagreement.

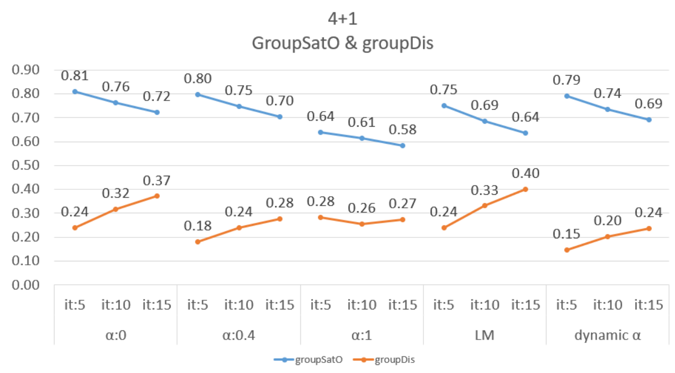


Figure 5: 4 similar – 1 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

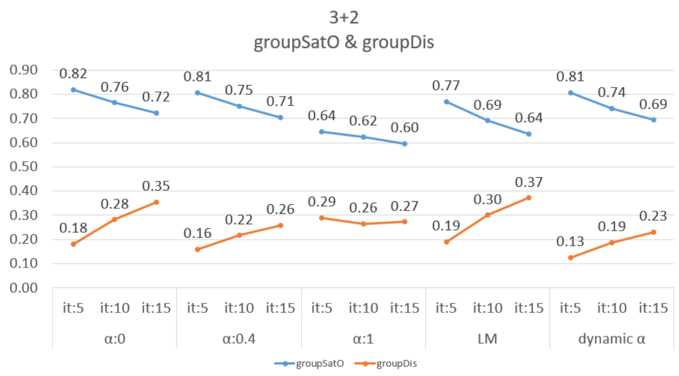


Figure 6: 3 similar – 2 similar: Overall group satisfaction and group disagreement for 100 groups for different μ.

Overall, the sequential group recommendations problem is not one dimensional. We need to examine the overall group satisfaction and the group disagreement in conjuncture with each other. Even though the average method ($\alpha = 0$) and the static value of $\alpha = 0.4$, offers slightly better group satisfaction than the dynamic α, they both have far higher group disagreement values. We argue that this is an acceptable loss of group satisfaction, when we consider the advantage that the dynamic α has over the average and static α aggregation methods on the disagreement between the group members.

To better demonstrate this, we also calculate the F-score of the *groupSatO* and *groupDis* values. We present these results in Table 1. For all group types and all number of iterations, the method using dynamic α offers better results, than the rest of the methods. This better demonstrates the point we made previously. Even if the average method offers better group satisfaction, our proposed dynamic hybrid sequential method, more than makes up for it, with the far lower group disagreement.

7 CONCLUSION

In this work, we introduce the notion of sequential group recommendation. We show that the standard group recommendation

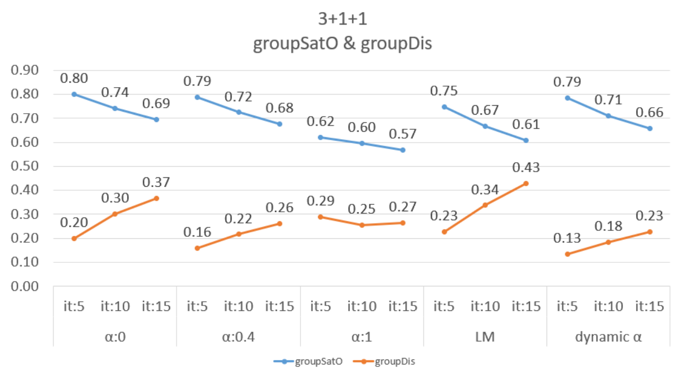


Figure 7: 3 similar – 1 dissimilar – 1 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

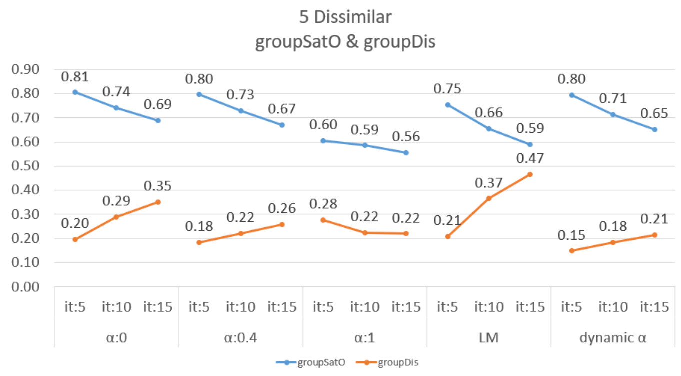


Figure 8: 5 dissimilar: Overall group satisfaction and group disagreement for 100 groups for different μ.

approaches are not favorable for sequential recommendations. Such methods suffer from particular drawbacks – a minority opinion may be lost by the average method, and the preference of the group is determined by just one voice in the least misery – that are only exacerbated when they are applied to sequential recommendations. We propose a sequential group recommendation model that takes into account the satisfaction of the members during the previous interactions of the group with the system. The influence that each member has on determining the group score for an item, is defined by the degree of the satisfaction that member has, for the recommended items in the previous iteration of the system. In the future, we want to further evaluate our work with a user study, to better examine the effectiveness of our method and how users respond to it. Additionally, we want to study the effect of sequential group recommendations on ephemeral groups and the subsequent consequence of them on the individual members. That is, how we can satisfy a user if at each iteration is a member of a different group, without sacrificing the overall satisfaction of the temporally formed group.

Group	**4+1**			**3+2**			**3+1+1**			**5Diss**		
Iterations	5	10	15	5	10	15	5	10	15	5	10	15
$\boldsymbol{\alpha = 0}$	0.784	0.720	0.672	0.819	0.741	0.682	0.801	0.719	0.662	0.805	0.726	0.668
$\boldsymbol{\alpha = 0.4}$	0.808	0.754	0.713	0.823	0.767	0.724	0.814	0.753	0.706	0.806	0.753	0.705
$\boldsymbol{\alpha = 1}$	0.675	0.673	0.648	0.676	0.676	0.655	0.662	0.663	0.640	0.658	0.667	0.649
LM	0.756	0.677	0.617	0.789	0.695	0.633	0.760	0.664	0.590	0.771	0.645	0.560
Dynamic $\boldsymbol{\alpha}$	**0.821**	**0.765**	**0.725**	**0.839**	**0.775**	**0.730**	**0.824**	**0.760**	**0.712**	**0.823**	**0.762**	**0.712**

Table 1: F-score values for all group categories per α value, per iteration.

ACKNOWLEDGEMENTS

This research was partially supported by the project *Ethical AI for the Governance of Society* (ETAIROS, grant #327352) funded by the Academy of Finland.

REFERENCES

[1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *PVLDB* 2, 1 (2009), 754–765.

[2] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group Recommendations with Rank Aggregation and Collaborative Filtering. In *RecSys*.

[3] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. 2011. Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-performance Recommender Systems. *ACM Trans. Web* 5, 1 (2011), 2:1–2:33.

[4] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval (SIGIR '18)*. https://doi.org/10.1145/3209978.3209998

[5] Lucas Augusto Montalvão Costa Carvalho and Hendrik Teixeira Macedo. 2013. Users' Satisfaction in Recommendation Systems for Groups: An Approach Based on Noncooperative Games. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13 Companion)*. https://doi.org/10.1145/2487788.2488090

[6] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-Interest Recommendation. In *International Joint Conference on Artificial Intelligence*.

[7] Zunping Cheng and Neil Hurley. 2009. Effective diverse and obfuscated attacks on model-based recommender systems. In *RecSys*.

[8] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized News Recommendation with Context Trees. *CoRR* abs/1303.0665 (2013).

[9] Francesca Guzzi, Francesco Ricci, and Robin Burke. 2011. Interactive Multi-party Critiquing for Group Recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. https://doi.org/10.1145/2043932.2043980

[10] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *RecSys*.

[11] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2015), 19:1–19:19.

[12] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys*.

[13] Anthony Jameson and Barry Smyth. 2007. The Adaptive Web. Springer-Verlag, Berlin, Heidelberg, Chapter Recommendation to Groups, 596–627.

[14] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *RecSys*.

[15] Kai Yu, A. Schwaighofer, V. Tresp, Xiaowei Xu, and H. . Kriegel. 2004. Probabilistic memory-based collaborative filtering. *IEEE TKDE* 16, 1 (2004), 56–69.

[16] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, and Young U. Ryu. 2010. A group recommendation system for online communities. *International Journal of Information Management* (2010). https://doi.org/10.1016/j.ijinfomgt.2009.09.006

[17] Defu Lian, Vincent W. Zheng, and Xing Xie. 2013. Collaborative Filtering Meets Next Check-in Location Prediction. In *WWW*.

[18] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *AAAI Conference on Artificial Intelligence*.

[19] Lucas Machado and Kostas Stefanidis. 2019. Fair Team Recommendations for Multidisciplinary Projects. In *WI*.

[20] Lihi Naamani Dery, Meir Kalech, Lior Rokach, and Bracha Shapira. 2010. Iterative Voting Under Uncertainty for Group Recommender Systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. https://doi.org/10.1145/1864708.1864763

[21] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. 2012. Fast Group Recommendations by Applying User Clustering. In *ER*.

[22] Eirini Ntoutsi, Kostas Stefanidis, Katharina Rausch, and Hans-Peter Kriegel. 2014. "Strength Lies in Differences": Diversifying Friends for Recommendations through Subspace Clustering. In *CIKM*.

[23] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. 2001. PolyLens: A recommender system for groups of user. In *Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*.

[24] Michael J. Pazzani and Daniel Billsus. 2007. *Content-Based Recommendation Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 325–341.

[25] D. Qin, X. Zhou, L. Chen, G. Huang, and Y. Zhang. 2018. Dynamic Connection-based Social Group Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018). https://doi.org/10.1109/TKDE.2018.2879658

[26] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (2018), 66:1–66:36.

[27] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*.

[28] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *CSCW*.

[29] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. *Collaborative Filtering Recommender Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 291–324.

[30] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in Package-to-Group Recommendations. In *WWW*.

[31] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. 2017. Fairness in Group Recommendations in the Health Domain. In *ICDE*.

[32] Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis. 2018. FairGRecs: Fair Group Recommendations by Exploiting Personal Health Information. In *DEXA*.

[33] Taghi M.; Su, Xiaoyuan;Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 2009 (2009). https://doi.org/10.1155/2009/421425

[34] John Von Neumann and Oskar Morgenstern. 1947. *Theory of games and economic behavior, 2nd rev. ed.* Princeton University Press.

[35] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *RecSys*.

[36] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: A Generative Model for Group Recommendation. In *KDD*.

PUBLICATION
II

Sequential group recommendations based on satisfaction and disagreement scores

Maria Stratigi, Evaggelia Pitoura, Jyrki Nummenmaa, and Kostas Stefanidis

J. Intell. Inf. Syst. 58.2 (2022), pp. 227–254
DOI: 10.1007/s10844-021-00652-x. URL: https://doi.org/10.1007/s10844-021-00652-x

Journal of Intelligent Information Systems (2022) 58:227–254
https://doi.org/10.1007/s10844-021-00652-x

Sequential group recommendations based on satisfaction and disagreement scores

Maria Stratigi[1] · Evaggelia Pitoura[2] · Jyrki Nummenmaa[1] · Kostas Stefanidis[1]

Received: 22 December 2020 / Revised: 8 June 2021 / Accepted: 16 June 2021 /
Published online: 14 July 2021

Abstract

Recently, group recommendations have gained much attention. Nevertheless, most approaches consider only one round of recommendations. However, in a real-life scenario, it is expected that the history of previous recommendations is exploited to tailor the recommendations towards meeting the needs of the group members. Such history should include not only which items the system suggested, but also the reaction of the members to these items. This work introduces the problem of sequential group recommendations, by exploiting the concept of satisfaction and disagreement. Satisfaction describes how well the group received the suggested items. Disagreement describes the satisfaction bias among the group members. We utilize these concepts in three new aggregation methods, SDAA, SIAA and Average+, designed to address the specific challenges introduced by sequential group recommendations. We experimentally show the effectiveness of our methods using big real datasets for both stable and ephemeral groups.

Keywords Group recommendations · Sequential recommendations · User satisfaction · User disagreement

✉ Maria Stratigi
maria.stratigi@tuni.fi

Evaggelia Pitoura
pitoura@cs.uoi.gr

Jyrki Nummenmaa
jyrki.nummenmaa@tuni.fi

Kostas Stefanidis
konstantinos.stefanidis@tuni.fi

[1] Tampere University, Tampere, Finland

[2] University of Ioannina, Ioannina, Greece

1 Introduction

Recommendations have become a standard in many services that target a consumer. From listening to music to health information, recommender systems are employed to make the user experience better and smoother. One of the most widely used methods to produce a user recommendation is the Collaborative Filtering (CF) method. Here, the system assumes that if two users had previously interacted with a set of items in a similar way, then the patterns of one user can be used to predict the other's future interactions.

With the expansion of social media, another form of recommendations has emerged; namely the group recommendations (Amer-Yahia et al., 2009; Ntoutsi et al., 2012; Ntoutsi et al., 2014). It is often the case that a group of users asks a recommender system for suggestions instead of a single user. A typical instance of group recommendations is the following. A group of friends wants to see a movie. Each friend has his/her preferences for what kind of movie he/she would like to watch. The recommender system needs to address these preferences adequately and report a set of items that have a degree of relevance to each group member. There is extensive research on the selection process of the items. Two approaches are the most popular. The first is to create a pseudo user by combining each group member's data and then applying a standard recommendation method. The second and most used approach is to apply a recommendation method to each member individually and then aggregate the separate lists into one for the group.

One of the most common requirement that the aggregation phase of a group recommender system needs to fulfill is *fairness* towards all the members of the group (Serbos et al., 2017; Stratigi et al., 2017; 2018; Machado & Stefanidis, 2019). A rudimentary definition of fairness is to recommend to the group the item that has the best relevance. Intuitively, a way to find such an item is to use the *Average* method on all group members' preference scores for it. In this case, all group members are considered equals. However, this method has a big drawback. For example, consider the following scenario with a group that consists of three users. Two of them have similar preferences to each other, while the third does not. The average method is prone to ignore the opinion of the last user. Another option is the *Least Misery* approach. Here, the user with the minimum preference score will act as a veto to the rest of the group. This method has a drawback as well. It only takes the opinion of one group member under consideration while ignoring the others. Subsequently, in most cases, the system recommends an item that is acceptable for all members, but it will be an item with a somehow *low* preference score.

In many works, a hidden assumption is that each time the group interacts with the recommender system is distinct from the rest. In a real-life scenario, however, the system needs to keep a history of the group's interactions to tailor its suggestion accordingly. Returning to the previous example, where the group meets every week for a movie night, the system cannot suggest the same list of items each time, regardless of whether those items have the best relevance for the group. Internally, the system needs to log all the items suggested to the group and the members' reaction to the proposed items (either with explicit feedback from the users or with an implicit method). To illustrate this requirement of group history, in our running example, one user has completely different preferences to the rest of the members. So each time the group meets, the same user is dissatisfied with the recommender system's suggestions. To remedy that, we introduce the notion of *sequential group recommendations* where the system considers the entire *sequence* of interactions for a group in order to produce its recommendations.

The concept of multiple rounds of recommendations allows us to make a conjecture. If a group member is dissatisfied with the proposed items at a specific iteration, he/she was either satisfied in the previous round or will be satisfied in the next. Subsequently, we exploit the notion of *satisfaction*, which estimates the degree of satisfaction in the recommendations for each group member, and *disagreement* which calculates the discrepancy among these satisfaction scores. We incorporate these satisfaction and disagreement scores during the group recommendation aggregation phase to ensure all members of a group are overall satisfied with the system's performance after several iterations.

Standard group recommendation approaches, such as Average and Least Misery, do not work very well when applied in a sequential scenario (Stratigi et al., 2020). Average produce good satisfaction scores but mediocre disagreement scores, while Least Misery produces reasonable disagreement and poor satisfaction scores. In this work, we propose three new aggregation methods.

The Sequential Dynamic Adaptation Aggregation, SDAA, method draws its inspirations from the individual advantages of the Average and Least Misery methods, while mitigating their drawbacks. It considers the group as a whole and dynamically at each iteration, estimates a weight based on the group members' satisfaction scores, to combine the equality of the Average method and the inclusivity of the Least Misery method. In contrast, the Sequential Individual Adaptation Aggregation, SIAA, method is user-centric. It calculates a weight for each member based on his/her satisfaction and disagreement scores, and applies this weight on the group members' preference scores during the aggregation phase. Finally, we propose a heuristic method, named Average+. Contrary to the others, which examine each item individually, Average+ considers the entire list of data items. It builds upon the average method to take advantage of the good satisfaction scores, but considers more than the requested k items. It incrementally adds items to the group recommendation list that produce the least disagreement scores when examined together with the rest of the list.

In a nutshell, the contributions of our work are the following:

– We introduce the problem of sequential group recommendations that targets to offer results that maximize the overall group satisfaction, and minimize the variance between the users satisfaction scores.
– We exploit the concept of *satisfaction* and *disagreement* for proposing three new aggregation methods designed to address the challenges of sequential group recommendations. Satisfaction describes each group member's degree of satisfaction for the items recommended at each iteration, and disagreement describes the degree of bias a user and the group faces at each iteration.
– We experimentally evaluate our methods, using two big real datasets, MovieLens and GoodReads, and test the methods for different types of groups. Furthermore, we study the performance of stable groups, where the group members are the same for all iterations, and ephemeral groups, where at the end of each iteration, a member of the group leaves, and a new one enters it.

A preliminary version of this work appears in Stratigi et al. (2020), where we introduced the SDAA method. In this article, we also propose the SIAA and Average+ methods. Furthermore, we evaluate the methods using a second real dataset, GoodReads, in addition to the MovieLens dataset used previously. Finally, we consider stable and ephemeral groups. In stable groups, the members remain the same throughout all the system's iterations. In contrast, after each iteration in ephemeral groups, a member of the group leaves, and a new one enters it.

The rest of the paper is structured as follows: Section 2 presents basic concepts on recommender systems, and Section 3 introduces the problem of sequential group recommendations. Sections 4, 5 and 6 describe the SDAA, SIAA, and Average+ aggregation methods, respectively. Section 7 presents our experimental setup, and Section 8 presents our evaluation results. Finally, Section 9 reports the related work, and Section 10 concludes this work with a summary of our contributions.

2 Basic concepts

In this section, we introduce a simple group recommendation process. It is a stand-alone process and does not consider any past interactions that the group may have had with the system. We use the group recommendation approach to aggregate the individual group members' preference lists into one group preference list. We define a preference list to be the list we recommend to each member individually or the group.

Specifically, let U be a set of users and I be a set of items. Each user $u_i \in U$ has given a rating from 1 to 5 to an item $d_z \in I$ represented as $r(u_i, d_z)$. The subset of users that rated an item $d_z \in I$ is denoted by $U(d_z)$, while the subset of items rated by a user $u_i \in U$ is denoted by $I(u_i)$. Let's assume also a group of users G of size n, $G \subseteq U$. Having applied a single user recommendation algorithm to all group members in G and produced their preference lists $A_{u_1}, \ldots, A_{u_n}$, the next step is to aggregate these lists into one. After the aggregation phase, each item will have just one group preference score and the top k will be reported back to the group as the final recommendations.

Typically, two well established aggregation methods are used (Baltrunas et al., 2010). The first is the *Average* method. The basic concept of this approach is that all members are considered equals. So, the group preference of an item will be given be averaging its scores across all group members: $avgG(d_z, G) = \frac{\sum_{u_i \in G} p(u_i, d_z)}{|G|}$, where $p(u_i, d_z)$ gives us the preference score of d_z for user u_i (computed by a standard single user recommendation algorithm). The second is *least misery* aggregation method, where one member can act as a veto for the rest of the group. In this case, the group preference score of an item d_z is the minimum score assigned to that item in all group members preference lists: $minG(d_z, G) = min_{u_i \in G}\, p(u_i, d_z)$.

3 Sequential group recommendations

In this section, we introduce the notion of multiple rounds to the previous group recommendation process. Consequently, we do not consider each group query to the system as a stand alone process, but as a sequence of queries submitted to the system by the same group. We call each group query to the system an *iteration*. Each iteration has the main components of a standard group recommendation method: single user recommendations followed by the aggregation phase. Formally, let $\mathcal{GR}$ be a sequence of μ group recommendations ($Gr_1, \ldots, Gr_\mu$). We use $p_j(u_i, d_z)$ for the preference score of user u_i for item d_z at iteration j, $1 \leq j \leq \mu$. A single user recommendation algorithm estimates these scores.

In contrast to the previous one, this new model introduces the notion of multiple *iterations* or *sequence* of recommendations. We use these interactions to modify the recommender system's output in such a way that if a user had a low satisfaction score in a previous iteration, he/she will likely have a higher one during the current iteration.

3.1 Satisfaction measure

To examine the effectiveness of our group recommender algorithm through a series of iterations, we need to define a measure that will help us elevate the problem from examining each iteration independently to examining a series of iterations. We introduce the notion of *satisfaction* that represents each group member's gratification for the recommended items after each iteration of the system. We define two variations of satisfaction: *single-user satisfaction* and *group satisfaction.*

For ease of reference, Table 1 holds all the symbols and notation used in the next sections.

3.1.1 Single user satisfaction

First, we want to provide a formal measure of the degree of each user's satisfaction, $u_i \in G$, to the group recommendation Gr_j received at step j. We do so, by comparing the quality of recommendations that the user receives as a member of the group with the quality of the user's recommendations as an individual. We consider the user's individual recommendation list, as his/her ground truth because to the best of the system's knowledge, this list is the only information about the user.

Let $A(u_i, j)$ returns a list with the top-k items for user u_i, that is, the k items with the highest prediction scores for u_i. Our goal is to directly compare the user's satisfaction from the group recommendation list with that user's ideal case. Doing so gives us a clearer view of the user's satisfaction than the average method since we consider the top items for each user, not only the top items for the group. Formally:

$$sat(u_i, Gr_j, j) = \frac{\sum_{d_z \in Gr_j} p_j(u_i, d_z)}{\sum_{d_z \in A(u_i, j)} p_j(u_i, d_z)}. \tag{1}$$

In the nominator, we calculate the user's satisfaction based on the group recommendation list. For every item in Gr_j, we sum the score as they appear in each user's top-k list. The denominator, calculates the ideal case for the user, by simply sum the scores of the k top items in the user's individual recommendation list. This way, we are able to normalize the user's satisfaction score. For example, if a user gives mainly low scores to items, then (1) can compensate for it.

Note that we do not use the scores as they appear in the group list, but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process somewhat distorts the group members' individual opinions, we opt to take into consideration only the personal preference scores of each group member.

Still, (1) remains *static*, in the sense that we calculate the satisfaction of a user for one iteration only. As stated before, what we want is to define a measure that considers the satisfaction scores from previous iterations of the system. Such a score will represent each group member's overall satisfaction with the entirety of the μ group recommendations.

Definition 1 (Overall Satisfaction) The overall satisfaction of user u_i with respect to a sequence $\mathcal{GR}$ of μ iterations is the average of the satisfaction scores after each iteration:

$$satO(u_i, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u_i, Gr_j, j)}{\mu}. \tag{2}$$

Table 1 The list of symbols and notations used in this paper

Notation	Description	Notation	Description
G	Group.	$satO(u_i, \mathcal{GR})$	The u_i's overall satisfaction for the sequence.
$\mathcal{GR}$	Sequence of iterations	$groupSat(G, Gr_j, j)$	Group satisfaction for Gr_j
μ	Number of iterations in the sequence	$groupSatO(G, \mathcal{GR})$	Overall group satisfaction for the sequence
Gr_j	Group recommendation list at iteration j	$userDis(u_i, G, Gr_j, j)$	u_i's disagreement in Gr_j at iteration j
$p_j(u_i, d_z)$	u_i's preference score for item d_z at iteration j	$groupDis(G, Gr_j)$	Group disagreement for the Gr_j.
$A(u_i, j)$	u_i's individual recommendation list	$groupDisO(G, gr)$	Overall group disagreement for the sequence
$sat(u_i, Gr_j, j)$	u_i's satisfaction score for Gr_j at iteration j		

3.1.2 Group satisfaction

Having defined each group member's satisfaction score, we can now define the satisfaction score of the entire group. Specifically, we define group G satisfaction concerning a group recommendation list Gr_j as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j, j) = \frac{\sum_{u_i \in G} sat(u_i, Gr_j, j)}{|G|}. \tag{3}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations $(Gr_1, \ldots, Gr_\mu)$, as:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u_i \in G} satO(u_i, \mathcal{GR})}{|G|}. \tag{4}$$

This measure indicates if the items we report to the group are acceptable to its members. Higher group satisfaction means that the group members are satisfied with the recommendations. However, since we average the members' satisfaction scores, a user's dissatisfaction can probably be lost in the computations. To counter this problem, we focus on the potential disagreements between the users in the group. For representing such *group disagreement* for just one instance of recommendations, we define the *groupDis* measure:

$$groupDis(G, Gr_j, j) = max_{u_i \in G} sat(u_i, Gr_j, j) - min_{u_i \in G} sat(u_i, Gr_j, j). \tag{5}$$

Subsequently, we define the overall group disagreement of G for the entire recommendation sequence as:

$$groupDisO(G, \mathcal{GR}) = max_{u_i \in G} satO(u_i, \mathcal{GR}) - min_{u_i \in G} satO(u_i, \mathcal{GR}). \tag{6}$$

Intuitively, we define the group's disagreement as the difference in the overall satisfaction scores between the most satisfied and the least satisfied member in the group. Ideally, we want this measure to take low values, indicating that the group members are all satisfied to the same degree. Higher $groupDis$ values will demonstrate that at least one member of the group is biased against.

To further portray our system's behavior, we define the *user disagreement* of a user to be the difference between the satisfaction score of the most satisfied group member and his/her satisfaction score. Formally:

$$userDis(u_i, G, Gr_j, j) = max_{\forall_{u_l \in G}} sat(u_l, Gr_j, j) - sat(u_i, Gr_j, j). \tag{7}$$

This allows us to determine better if a group member is systematically being favored (he/she will have very low user disagreement scores) or is disregarded (he/she has very high user disagreement scores).

3.2 Problem definition

Our sequential group recommender system needs to achieve two independent objectives that are critical to the success of the model. The first objective considers the group as an entity:

we want to offer the best possible results for the group. The second objective considers the group members independently: we want to behave as fairly as possible towards all members.

Definition 2 (Sequential Group Recommendations) Given a group G, the sequential group recommender produces at the j^{th} iteration a list of k items Gr_j, $Gr_j \in \mathcal{GR}$, that:

1. Maximizes the overall group satisfaction, $groupSatO(G, \mathcal{GR})$, and
2. Minimizes the overall variance between users satisfaction scores: $groupDisO(G, \mathcal{GR})$.

To achieve the first objective, we target to maximize the $groupSatO$ value. For that, we require the items with the highest preference scores for the group as a whole. Since $groupSatO$ expresses the average satisfaction of the group members, and the dissatisfaction of just one member is easily lost, we also need to achieve the second objective, to minimize $groupDisO$, which represents the degree of dissatisfaction between the members of the group.

Depending on the similarity between the group members, these two objectives may be conflicting with each other. A simple example is a group of three members, two are highly similar to each other, and one is very dissimilar to them. To achieve high $groupSatO$ values, we need to recommend items relevant to the two similar users to increase the average satisfaction of the group. On the other hand, by doing so, $groupDisO$ will take high values since we do not address the needs of the third member. Overall, we need to recommend items that are not just good enough for all members, since that still returns low $groupSatO$ values, but items that have high relevance to the group, without sacrificing the opinions of the minority.

4 Sequential dynamic adaptation aggregation method

Generally speaking, both the Average and the Least Misery aggregation methods have drawbacks when we consider them for sequential recommendations. Simultaneously, both have advantages (namely, equality for Average and inclusion of all opinions for Least Misery) that are assets for a recommender. Furthermore, it has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem (Xiao et al., 2017). We want to achieve that throughout the multiple iterations of the system, to be equally fair to all members of the group.

The first method we examine, capitalizes on the advantages of the Average and the Least Misery aggregation methods and proposes a novel aggregation method that is a weighted combination of them. We call the method *sequential dynamic adaptation aggregation method*, or shortly SDAA. The preference score of an item for a group in SDAA is computed as follows:

$$score(G, d_z, j) = (1 - \alpha_j) * avgG(G, d_z, j) + \alpha_j * leastG(G, d_z, j). \tag{8}$$

The function $avgG(G, d_z, j)$ returns the score of the item d_z as it is computed by the average aggregation method during iteration j, and function $leastG(G, d_z, j)$ returns the least satisfied user's score of d_z at iteration j. The variable α takes values from 0 to 1. If $\alpha = 0$, then the sequential hybrid aggregation method becomes average aggregation. At the same time, when $\alpha = 1$, it transforms to a modified least misery aggregation, and we only consider the preferences of the least satisfied member.

Intuitively, when $\alpha = 0$, our method satisfies the first part of the fair sequential group recommendation problem since the average aggregation considers the best options for the group as a whole. The benefits of $\alpha = 1$ can be seen for higher values of μ. At the first iterations, the group disagreement may remain high since it considers the users' overall satisfaction (2).

Given that our goal is to fulfill both the problem definition's objectives, we need to set the value of α between 0 and 1. Furthermore, we want this value to self-regulate, to describe the consensus of the group more effectively. Thus, we set the value of α dynamically in each iteration by subtracting the group members' minimum satisfaction score in the previous iteration from the maximum score.

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}, j-1) - min_{u \in G} sat(u, Gr_{j-1}, j-1), \tag{9}$$

where $j > 1$. When $j = 1$ (the first iteration of the system), then $\alpha = 0$, and the aggregation method reverts to that of a classic average aggregation.

By having this dynamically calculated α, we counteract the average and least misery's drawbacks. Intuitively, suppose the group members are equally satisfied at the last round. In that case, α takes low values, and the aggregation will closely follow that of an average, where everyone is treated as an equal. On the other hand, if one group member is extremely unsatisfied in a specific iteration, α takes a high value and promotes that member's preferences on the next iteration. Formally:

Property 1 Let u be a user in a group G, such that,
$sat(u, Gr_{j-1}, j-1) < sat(u_i, Gr_{j-1}, j-1)$, $\forall u_i \in G \backslash \{u\}$. Then, $\exists u_y \in G \backslash \{u\}$, such that, $sat(u_y, Gr_j, j) < sat(u, Gr_j, j)$.

Proof For the purpose of contradiction, assume that $sat(u, Gr_j, j) < sat(u_y, Gr_j, j)$, $\forall u_y \in G \backslash \{u\}$. Then, this means that u is the least satisfied user at iteration j, which in turn means that α takes values not leading towards a least misery approach at this iteration, meaning that u was not the least satisfied user at iteration $j-1$, which is a contradiction. □

The SDAA method is described in Algorithm 1. The input to the algorithm is the group G, the iteration j, and the size k of the group recommendation list Gr_j, which we report to the group after each iteration is finished. n Line 1 we construct a set, Gl, that contains all the items that appear in the group members individuals recommendation lists. In Lines 2–6, we calculate the α; if it is the first iteration, meaning $j = 1$ (Line 2), then $\alpha = 0$, otherwise, we use (9) (Line 5) to calculate it. In Lines 7–10, we perform the SDAA method. For all the items in Gl, we calculate the item score using (8) (Line 8) and insert them in the group list Gr_j (Line 9). After we have examined all the items in Gl, we sort the items in the group list (Line 11), and finally, we report the top-k of them to the group (Line 12). The complexity of our algorithm is $\mathcal{O}(n \log n)$ due to the time complexity of the sorting function, where $n = |Gl|$.

Algorithm 1 SDAA aggregation algorithm.

```
   Data: Group G, iteration $j$, top $k$
   Result: Group Recommendation List: $Gr_j$
1  $Gl \leftarrow \cup_{u_i \in G} A(u_i, j)$;
2  if $j$=1 then
3      $\alpha_j = 0$;
4  else
5      $\alpha_j = maxSat(G, j-1) - minSat(G, j-1)$;
6  end
7  for item $d_z$ in $Gl$ do
8      $score(G, d_z, j) = (1-\alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j)$;
9      $Gr_j \leftarrow Gr_j \cup score(G, d_z, j)$;
10 end
11 $sort(Gr_j)$;
12 $report(top(Gr_j, k))$;
```

5 Sequential individual adaptation aggregation method

The goal of the SDAA method is to calculate a weight score in order to combine the average and least misery methods. This weight was the same for all group members, and even though it was calculated using all members' satisfaction scores, we did not directly address each individually. Thus a member's interest can be ignored. As a result of this observation, we propose a new aggregation method that concentrates on each group member individually. We accomplish that by assign each member a *weight* score. This weight is based on each user's statistics, namely the satisfaction and disagreement scores, to calibrate the system for each user independently.

Specifically, during the group recommender's aggregation phase, we exploit the concept of overall user satisfaction and user disagreement. Each group member is assigned an individual weight, which is applied to an item's relevance score given by the recommender system (10). The final score of that item for the group is the summation of its weighted relevance score for all group members.

$$score(G, d_z, j) = \sum_{u_i \in G} w_{u_i, j} * p_j(u_i, d_z). \tag{10}$$

Taking into consideration the particulars of our problem, we pay attention to satisfaction and disagreement values. Since we want them to be applied to each group member individually, we do not use the group variant of these measures, but instead, we focus on the individual ones (2 and 7).

The satisfaction score of a user offers a way to balance the recommendations for all members of a group through a series of recommendation iterations. These members that were not satisfied in the previous iterations are promoted for the next ones. A straightforward approach to accomplish this is to utilize the overall satisfaction of a user. We want the users with a low overall satisfaction score to have a higher weight compared to those with a high overall satisfaction score. More specifically, let assume that the group is in the j-th iteration. We calculate the overall satisfaction score for all $j-1$ iterations (2). Subsequently, the weight that is assigned for each member at iteration j is:

$$w_{u_i,j} = 1 - satO(u_i, \mathcal{GR}_{j-1}). \tag{11}$$

where $\mathcal{GR}_{j-1}$ are all previous recommendations at the $j-1$ iterations of the system.

The users with high overall satisfaction are assigned a lower weight, while the users with lower satisfaction scores are assigned a higher one. If the group members are satisfied to the same degree, then they are assigned similar weights. Since we compute a user's overall satisfaction score, we ensure that a user that is systematically biased against will have the highest weight.

Such an aggregation method is based only on the user's overall satisfaction score, which is his/her average satisfaction scores. As with any Average approach, it makes the aggregation method slow to react in extreme cases. Suppose a user was highly satisfied in the first μ iterations and then extremely dissatisfied in the rest. In that case, the method will require some iterations to assign him/her a higher weight to promote him/her. The same is also true in the reverse case, where a user was dissatisfied and then became highly satisfied. This leads to an aggregation method that promotes him/her even though he/she is satisfied during some iterations.

A way to counter this drawback, is to shorten the iterations needed for a change in user's satisfaction to be reflected in the user's assigned weight. To achieve this, we introduce the *sequential individual adaptation aggregation method*, or SIAA, that considers not only the overall satisfaction of a user, but also the user disagreement of the previous iteration.

$$w_{u_i,j} = (1-b) * (1 - satO(u_i, \mathcal{GR}_{j-1})) + b * userDis(u_i, G, Gr_{j-1}, j-1), \tag{12}$$

where b is the weight we use to balance the overall satisfaction and user disagreement scores. This is a constant variable and we found experimentally its best value. Given that we are in the j-th iteration of the system, $\mathcal{GR}_{j-1}$ expresses the recommendations of all the previous $j-1$ iterations. Finally, Gr_{j-1} includes the recommendations of the immediately preceding iteration.

To generalize, if a user was severely dissatisfied in the previous iteration and is overall dissatisfied, then the weight is high. If the user was dissatisfied in the previous round but overall has a high satisfaction score, then the weight is significantly lower. This is in accordance with the previous method that disregards any disagreements. However, the added variable of the user disagreement influences the next iteration. Specifically, suppose the user was considerably biased against in the current iteration (has a low satisfaction score while another member has a high one). In that case, he/she will be promoted faster than the plain satisfaction weighted method. Furthermore, if a user is the most satisfied in the current iteration, then his/her weight will be lowered to promote alternative group members.

The SIAA method is described in Algorithm 2. The output and Line 1 are the same as Algorithm 1. Additionally, we have the same input as 1 but with the added variable b, which is used to calculate the weight score (Line 5). This calculated weight is then applied to group members' preference score that the recommender system has predicted for each item d_z, denoted as $p_j(u_i, d_z)$, $u_i \in G$, $d_z \in Gl$. The final score of the item is the weighted sum for all users (Line 6). Finally, as in Algorithm 1, we sort the group recommendation list Gr_j and report to the group the top k items (Lines 10-11). The complexity of our algorithm is $\mathcal{O}(n \log n)$, due to the time complexity of the sorting function, where $n = |Gl|$.

Algorithm 2 SIAA aggregation algorithm.

```
   Data: Group G, iteration $j$, top $k$s, $b$
   Result: Group Recommendation List: $Gr_j$
1  $Gl \leftarrow \cup_{u_i \in G} A(u_i, j)$;
2  for item $d_z$ in $Gl$ do
3      $score(G, d_z, j) = 0$;
4      for $u_i$ in $G$ do
5          $w_{u_i,j} = (1-b) * (1 - satO(u_i, \mathcal{GR}_{j-1})) + b * userDis(u_i, G, Gr_{j-1})$;
6          $score(G, d_z, j) = score(G, d_z, j) + w_{u_i,j} * p_j(u_i, d_z)$
7      end
8      $Gr_j \leftarrow Gr_j \cup score(G, d_z, j)$;
9  end
10 $sort(Gr_j)$;
11 $report(top(Gr_j, k))$;
```

6 Average+ aggregation method

All the previous aggregation methods examine how good a *single* item is for the group and add it in the group recommendation list, without taking into account the items already present there. Since we examine the users' satisfaction based on the entire list, adding to the list each item individually may not produce the best suggestions. Based on this observation, we propose the Average+ aggregation method that exploits the Average method that offers the best group satisfaction (Stratigi et al., 2020). Specifically, Average concentrates on finding the items that will satisfy the majority of the group. At the same time, Average also has high group disagreement scores because it is prone to ignore the outlier of a group. If one group member is dissimilar to the rest of the group, due to the Average method, he/she is almost always dissatisfied. This observation drives us to propose a new aggregation method, which not only secures the advantages of the Average method but, at the same time, mitigates the problems that the drawback mentioned above creates.

This aggregation method has two phases. In the first phase, we capitalize on the advantage of the average method, and, in the second phase, we counter its drawback. The first phase is straight-forward. The average aggregation method does all the work for us. Since it is the method with the best group satisfaction scores, we use a standard average aggregation method for the group. However, we do not take the last step of a group recommender process: a group recommendation system produces a long list of potential items but it only presents to the group the top ones with the highest predicted score. Instead of only keeping that short list, we keep a significantly longer one that contains the items with the highest scores.

After experimentation, the size of the list that provides the best results is $5k$, where k is the number of items we propose to the group. We denote this list as $AvgList_j^G$ for group G at iteration j. We do not want to take a longer list since the items further down typically do not have good enough group relevance scores (meaning that the items are not relevant to most of the group), and hence are not good enough for us to use to take advantage of the performance of the average aggregation method.

Having found the most relevant items for the group and subsequently have secured a high group satisfaction, we next want to minimize the group disagreement. We first exploit a simple idea. For each item in the $AvgList_j^G$ list, we calculate the group disagreement score they would have generated if they were the only item proposed to the group. Meaning that Gr_j consists of only that item (13). We return to the group the k ones with the lowest group disagreement.

$$Gr_j = Gr_j \cup \{d_z \mid min_{\forall d_z \in AvgList_j^G}(groupDis(G, d_z, j)) \ \vee d_z \notin Gr_j\}. \tag{13}$$

This allows us not only to narrow our search to items that are relevant to the group, but at the same time ensure that they have the lowest group disagreement possible.

This is a simple idea because it examined each item individually. However, the system returns to the group a list of items, and we calculate the various satisfaction and disagreement scores based on the entire list. We consider the list as a whole, not each item independently. For example, a user may not be interested in one item but be excited for another. These two items can balance each other out and offer better group satisfaction and disagreement scores. Therefore, we propose a more sophisticated method.

To find the best k items to propose to the group based on the group satisfaction and group disagreement scores they generate is a hard problem. It has been proven that to compute the best suggestions for a group that minimizes the gap between the least and highest satisfied group members is an NP-Hard problem (Xiao et al., 2017).

Thus, we propose an incremental heuristic method. This method consists of two steps. In the first step, we include in the group recommendation list Gr_j the item with the highest predicted score in $AvgList_j^G$. The second step of the method is recursive. At each step, we examine one by one all the items in the $AvgList_j^G$. We include in the list the item that its inclusion generates the lowest group disagreement score (14). We perform this step until we have included k items in the group recommendation list.

$$Gr_j = Gr_j \cup \left\{d_z \mid min_{\forall d_z \in AvgList_j^G}(groupDis(G, Gr_j \cup d_z, j)) \vee d_z \notin Gr_j\right\}. \tag{14}$$

By incrementally filling the group recommendation list, we can examine all the items and the effects they collectively have for the group.

We describe this process in Algorithm 3. As input, we have the group G, the iteration j of the system, the number k of the reported items, and the list $AvgList_j^G$, which contains the $5k$ items with the highest predicted scores after applying the average aggregation method. The output of the algorithm is the group recommendation list Gr_j. In Line 1, we insert in the group recommendation list Gr_j the item with the highest predicted score given by the average aggregation. The recursion starts in Line 2 and stops when we have included in the Gr_j the requested k items. In Line 3 we iterate over all the items in the $AvgList_j^G$ list and select the item d_z that produces the minimum group disagreement score and it is not already included in the group recommendation list Gr_j. In Line 4 we insert the selected item to the list. Finally, we report to the group the recommendation list. The complexity of this algorithm is $\mathcal{O}(n^2)$ where $n = |Gr_j|$. We perform the inner loop n times, over $5n$ items.

Algorithm 3 Average+ aggregation algorithm.

```
   Data: Group G, iteration $j$, top $k$, $AvgList_j^G$
   Result: Group Recommendation List: $Gr_j$
1  $Gr_j \leftarrow AvgList_j^G.getFirst()$;
2  while $|Gr_j| < k$ do
3      select an item $d_z \in AvgList_j^G \setminus Gr_j$ that minimizes: $groupDis(G, Gr_j \cup d_z, j)$;
4      Add item $d_z$ to $Gr_j$: $Gr_j = Gr_j \cup d_z$;
5  end
6  $report(top(Gr_j, k))$;
```

7 Experimental setup

7.1 Datasets

For the experimental evaluation of our sequential group recommender system, we do not have access to any dataset that contain ratings for groups. We instead artificially created groups using two real datasets, namely the 20M MovieLens Dataset (Harper & Konstan, 2015), and the 15M book reviews from GoodReads (Wan et al., 2019). MovieLens contains 20M ratings across 27,3K movies given by 138,5K users between 01.1995 and 03.2015. The GoodReads dataset contains around 15M ratings from 465K users for about 2M books. To simulate multiple iterations of suggestions, we split the datasets into chunks. Each chunk is added to the system, representing new information, and along with the already existing data in the system, is used for locating the suggestions for the next iteration.

MovieLens dataset Initially, we divide the dataset into two parts of roughly the same size. The first part that contains the ratings given between 01.1994 and 12.2003, is the starting dataset of the recommender. This gives us an initial dataset that consists of 8.381.255 ratings, 73.519 users and 6.382 movies. We initiate the system with this starting dataset to avoid any issues emanating from the cold start problem.[1] The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of 6 months. During the first iteration, the system will have access only to the initial dataset. When that iteration ends, the system will be enhanced with one additional chunk, and after each subsequent iteration, the system will be given access to the next chunk.

GoodReads dataset For GoodReads, we altered the splitting process due to the difference in the ratings' distribution over time. The distribution is skewed towards the latter years, with the starting years having very few ratings. The starting data for MovieLens corresponded to around 40% of the entire dataset. We take a comparable size for GoodReads, which amounts to around 6.296.000 ratings, and split the rest into equal-sized chunks that

[1]The cold start problem in collaborative filtering appears when there is not enough information about a user and we are unable to find similar other users to him/her. This problem is beyond the scope of this research, and to overcome it, we initialize our system with a large enough dataset.

contain 674.565 additional ratings each. It is worth noting that GoodReads is far more sparse than MovieLens. Subsequently, this is going to have an adverse effect on the quality of the recommendations.

7.2 Group formation

The experimental procedure of (Stratigi et al., 2020) was focused on stable groups. This means that the group members do not change between iterations. We extend the experiments for this work to also include ephemeral groups. We now allow group members to leave the group and new ones to enter it. The users that have left a group are allowed to participate in the group again.

We examine our recommender on three types of groups based on the similarity shared between the members. We calculate the similarity between the group members, employing the starting dataset, using Pearson Correlation (Resnick et al., 1994), which takes values from -1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity:

$$s(u_i, u_l) = \frac{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})(r(u_l, d_z) - \bar{r}_{u_l})}{\sqrt{\sum\limits_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})^2} \sqrt{\sum\limits_{d_z \in X} (r(u_l, d_z) - \bar{r}_{u_l})^2}}, \text{ where } X = I(u_i) \cap I(u_l)$$

and $\bar{r}_{u_i}$ is the mean of the ratings in $I(u_i)$, i.e., the mean of u_i's ratings.

We consider two users to be highly similar to each other if they share an above 0.5 similarity score, while dissimilar when they have -0.5 or lower similarity score. The types of groups we are considering are the following:

4 similar – 1 dissimilar (4+1): The four members of the group share a high similarity score, while the dissimilar one shares a low similarity score with the rest of the group members.

3 similar – 2 similar (3+2): We divide the group into two subgroups. The members of each subgroup are similar to each other, while at the same time, the subgroups are dissimilar to one another, i.e., all members of one subgroup are dissimilar to all members of the other subgroup.

5 dissimilar: All members of the group are dissimilar with each other.

With these group formats, we want to simulate three different real-life scenarios. First, when a new person enters an established group, and his/her interests may very well be dissimilar to the rest of the group. For example, consider a working environment in which a new colleague joins an existing project team. Second, when two different groups join together for a single activity, such as two working groups that need to collaborate on a new project. The final group type simulates the case when random people join together for an activity, like in a business lunch or a tour offered by a travel agency.

We will also examine the performance of our aggregation methods for ephemeral groups. The initial members of the group will be the same as the stable ones. To simulate the ephemeral groups' dynamic characteristics, we randomly remove a group member after each iteration and randomly include a new user in the group so as the group size to remain the same. The new member selected is such that the type of the group does not change. For example, if we have a 4+1 group, and the one dissimilar user is removed, then the new member has to be dissimilar to the four remaining group members.

8 Experiments

8.1 Experimental procedure

In our experiments, we examine the behavior of the three types of groups. For each type, we consider 100 different groups with five members. For all the groups in the set, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDisO(G, \mathcal{GR})$ at the end of each iteration. In the ephemeral groups, at each iteration, we focus only on the users that are currently members of the group. Additionally, to examine the overall performance of the aggregation methods, we utilize the harmonic mean of $groupSatO$ and $groupDisO$, namely their F-score. Considering the input functions that F-score needs, we use $1 - groupDisO$ to simulate the group agreement. $F\text{-}score = 2\frac{groupSatO*(1-groupDisO)}{groupSatO+(1-groupDisO)}$.

To further analyze the methods' quality, we use the Normalized Discounted Cumulative Gain (NDCG), where we want the best possible items for a user to appear with the highest ranking possible in the group recommendations:

$NDCG(u_i, G, j) = \frac{DCG(u_i,G,j)}{IDCG(u_i,G)}$, with $DCG(u_i, G, j) = \sum_{d_z \in Gr_j} \frac{|d_z \cap A(u_i,j)|}{log_2(r(d_z,Gr_j)+1)}$, where $r(d_z.Gr_j)$ is the rank of item d_z in the group recommendation list Gr_j. IDCG is the best possible outcome for user u_i. We supplement the results from NDCG, with the Discounted First Hit (DFH) metric that counts if a user has seen an item that is highly relevant to him/her in the early ranks of the group recommendation list: $DFH(u_i, Gr_j) = \frac{1}{log_2(fh+1)}$, where fh is the rank of the first item that appears both in the group recommendation list and the individual preference list of the user. Since NDCG and DFH refer to one user, we apply them to the group by computing them for each member and then taking the average over them. We do this at the end of each iteration.

We find similarities between users by utilizing the Pearson Correlation. We consider two users as similar if the similarity is greater than a threshold and if they have rated more than 5 identical items. Due to the difference in each dataset's semantics, the similarity threshold we set for the MovieLens dataset is 0.8, and for the GoodReads dataset is 0.7. To predict preference scores for a user, we use the Weighted Sum of Others Ratings (Su & Khoshgoftaar, 2009), and take into account only the top 100 most similar users (denoted as P_{u_i}) to him/her: $p(u_i, d_z) = \bar{r}_{u_i} + \frac{\sum_{u_l \in (P_{u_i} \cap U(d_z))} s(u_i,u_l)(r(u_l,d_z)-\bar{r}_{u_l})}{\sum_{u_l \in (P_{u_i} \cap U(d_z))} |s(u_i,u_l)|}$. We recommend to the group the 10 items with the highest group preference score that are not previously recommended.

For the ephemeral groups, we simultaneously perform all aggregation methods with the same group. Since the members' exit and entry is done randomly, this ensures that the experiment is as fair as possible for all different aggregation methods.

We consider six different aggregation methods.

- **Avg** is the classic Average aggregation method.
- **RP80** (Amer-Yahia et al., 2009) combines group relevance and group disagreement scores. As group relevance it utilizes the average method and as a group disagreement method it uses the Average Pair-wise Disagreement:
  $dis(d_z, G) = \frac{2}{|G|(|G|-1)} \sum(|p(u_i, d_z) - p(u_l, d_z)|)$, where $u_i, u_l \in G$, and $u_i \neq u_l$. The Average Pair-wise Disagreement reflects the degree of consensus in the relevance scores for d_z among group members. The final score for item d_z for the group G is: $\mathcal{F}(d_z, G) = (1-w) * avg(d_z, G) + w * dis(d_z, G)$. We choose to set variable w to 0.8, since it performs better based on the experiments presented in Amer-Yahia et al. (2009).

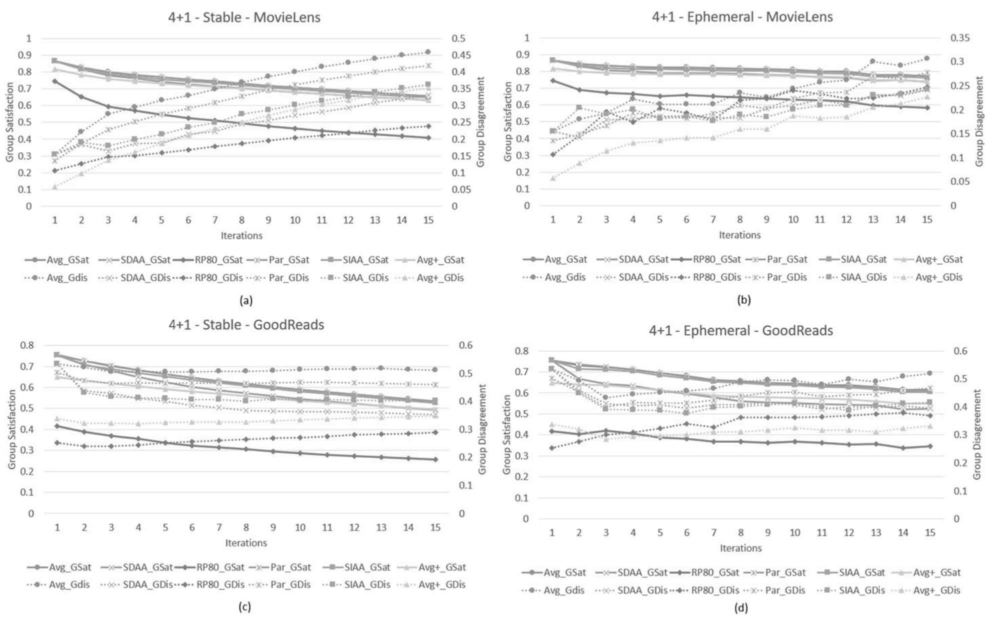


Fig. 1 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads

– **Par** (Xiao et al., 2017) generates a group recommendation list by incrementally adding to it items that have the best combination of Social Welfare (SW) and Fairness (F) scores. The Social Welfare of an item is the average satisfaction of all members for

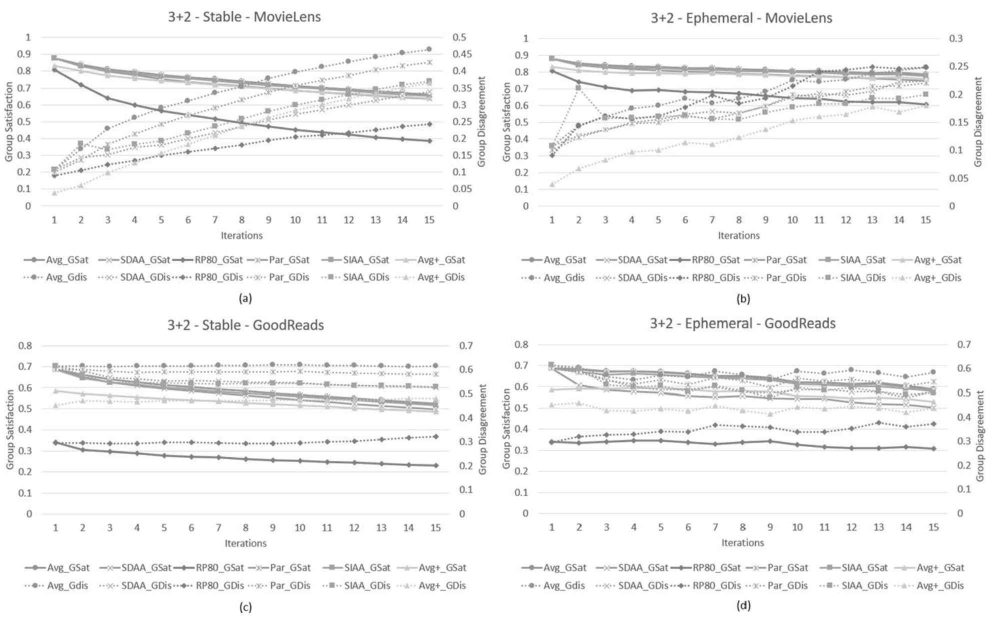


Fig. 2 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads

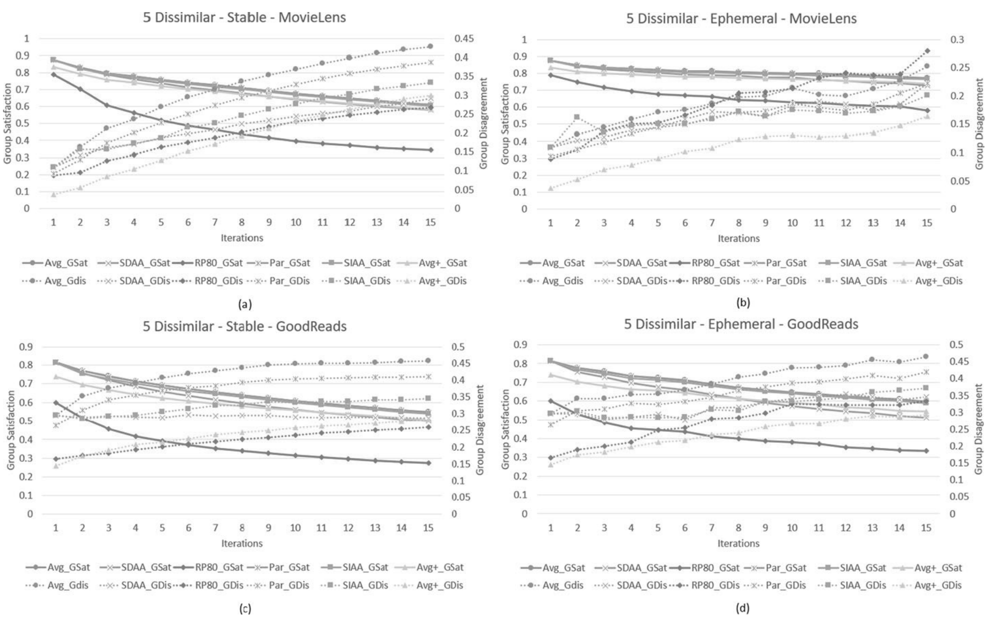


Fig. 3 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for stable and ephemeral groups, using MovieLens and GoodReads

that item. As Fairness it employs the variance over the member's satisfaction. The final score of an item for the group is: $\mathcal{F}(d_z, G) = \lambda * SW(d_z, G) + (1 - \lambda) * F(d_z, G)$, where $\lambda = 0.8$ according to (Xiao et al., 2017).

- **SDAA** is the Sequential Dynamic Adaptation Aggregation method proposed in this work.
- **SIAA** is the Sequential Individual Adaptation Aggregation method proposed in this work.

Table 2 F-score for all aggregation methods for both stable and ephemeral groups

	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
4+1 Groups / MovieLens Dataset												
1	0.855	0.855	0.812	*0.865*	0.855	**0.875**	0.855	0.855	0.812	*0.865*	0.855	**0.875**
5	0.724	**0.774**	0.660	0.746	*0.771*	0.770	0.805	0.802	0.716	*0.817*	0.815	**0.821**
10	0.649	**0.704**	0.585	0.671	*0.697*	0.695	0.781	0.783	0.690	0.791	**0.798**	*0.792*
15	0.594	**0.650**	0.532	0.616	*0.643*	0.640	0.732	0.748	0.657	0.746	**0.760**	*0.756*
4+1 Groups / GoodReads Dataset												
1	0.576	0.576	0.535	*0.599*	0.576	**0.655**	0.576	0.576	0.535	*0.599*	0.576	**0.655**
5	0.567	0.612	0.465	0.592	*0.619*	**0.631**	0.613	0.602	0.493	0.635	*0.646*	**0.655**
10	0.533	0.587	0.412	0.559	*0.588*	**0.595**	0.567	0.572	0.466	0.591	*0.611*	**0.622**
15	0.510	*0.559*	0.378	0.536	*0.559*	**0.562**	0.539	0.561	0.446	0.569	*0.593*	**0.602**

With bold is the best value and with italics the second best

Table 3 F-score for all aggregation methods for both stable and ephemeral groups

	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
3+2 Groups / MovieLens Dataset												
1	0.885	0.885	0.855	*0.889*	0.886	**0.891**	0.885	0.885	0.855	*0.889*	0.886	**0.891**
5	0.743	0.784	0.678	0.767	*0.787*	**0.790**	0.826	0.827	0.759	*0.840*	0.831	**0.844**
10	0.654	**0.706**	0.575	0.676	0.702	*0.700*	0.790	0.791	0.708	0.803	**0.810**	*0.809*
15	0.592	**0.648**	0.510	0.614	*0.641*	0.637	0.768	0.762	0.673	*0.782*	**0.787**	**0.787**
3+2 Groups / GoodReads Dataset												
1	0.493	0.493	0.458	*0.506*	0.493	**0.565**	0.493	0.493	0.458	*0.506*	0.493	**0.565**
5	0.473	0.510	0.398	0.490	*0.519*	**0.537**	0.522	0.526	0.454	0.537	*0.548*	**0.578**
10	0.455	0.493	0.371	0.473	*0.502*	**0.521**	0.495	0.515	0.436	0.516	*0.538*	**0.558**
15	0.444	0.486	0.343	0.466	*0.493*	**0.502**	0.487	0.498	0.413	0.512	*0.537*	**0.546**

With bold is the best value and with italics the second best

– **Avg+** is the Average+ Aggregation method proposed in this work.

8.2 Experimental results

In Figs. 1, 2 and 3, we show the group satisfaction and group disagreement scores for the varying group types for 15 iterations of the system. We want to achieve high group satisfaction and low group disagreement scores. In Tables 2, 3 and 4, we depict their corresponding F-scores. In Tables 5 and 6 we present the NDCG and DFH scores for each aggregation method, for both stable and ephemeral groups, in the MovieLens and GoodReads datasets. We calculate the average scores for all iterations of the system for each aggregation method.

Table 4 F-score for all aggregation methods for both stable and ephemeral groups

	Stable Groups						Ephemeral Groups					
It	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
5 Dissimilar Groups / MovieLens Dataset												
1	0.882	0.882	0.846	*0.891*	0.882	**0.893**	0.882	0.882	0.846	*0.891*	0.882	**0.893**
5	0.745	0.774	0.641	0.766	*0.782*	**0.789**	0.824	0.829	0.751	*0.837*	0.831	**0.843**
10	0.653	0.695	0.522	0.674	*0.696*	**0.700**	0.793	0.795	0.699	0.806	*0.808*	**0.815**
15	0.591	**0.637**	0.466	0.611	0.633	*0.636*	0.759	0.752	0.643	0.774	*0.781*	**0.784**
5 Dissimilar Groups / GoodReads Dataset												
1	0.755	0.755	0.698	*0.773*	0.755	**0.793**	0.755	0.755	0.698	*0.773*	0.755	**0.793**
5	0.639	0.684	0.526	0.660	*0.687*	**0.696**	0.682	0.690	0.561	0.699	*0.714*	**0.716**
10	0.580	0.627	0.445	0.603	*0.632*	**0.637**	0.605	0.613	0.487	0.629	*0.653*	**0.655**
15	0.546	0.589	0.401	0.568	*0.592*	**0.595**	0.566	0.574	0.445	0.589	*0.608*	**0.615**

With bold is the best value and with italics the second best

Table 5 NDCG and DFH scores for the stable groups

	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
MovieLens Dataset / Stable Groups												
4+1	0.042	**0.044**	0.020	0.042	*0.043*	0.034	0.894	0.861	0.803	*0.906*	0.889	**0.933**
3+2	0.042	**0.044**	0.020	*0.043*	*0.043*	0.035	0.905	0.867	0.809	*0.918*	0.898	**0.939**
5 Dis	0.041	**0.042**	0.017	*0.042*	0.041	0.034	0.892	0.838	0.780	*0.909*	0.885	**0.934**
GoodReads Dataset / Stable Groups												
4+1	0.055	**0.062**	0.014	*0.057*	*0.057*	0.045	0.829	0.759	0.696	*0.847*	0.826	**0.887**
3+2	0.053	**0.057**	0.013	*0.055*	*0.055*	0.042	0.784	0.742	0.607	*0.796*	0.782	**0.829**
5 Dis	0.057	**0.062**	0.018	*0.058*	*0.058*	0.047	0.861	0.777	0.766	*0.877*	0.859	**0.919**

With bold is the best value and with italics the second best

8.2.1 Stable groups

For the stable groups, we see from the overall performance of the methods (F-scores) that the SDAA method outperforms the classic Average method for all group types. SDAA, SIAA, and Average+ are the best three performing methods for all group types for both datasets.

In the MovieLens dataset, for the more homogeneous group types (4+1 and 3+2), SDAA performs slightly better in the latter iterations, with our new proposed methods SIAA and Average+ being a close second. The differences in performance between SDAA, SIAA, and Average+ are very low for all group types. This especially apparent in the latter iterations, where the differences in the scores are around 2%.

For the GoodReads dataset, Average+ is the most effective method for all group types. Average+ offers the best group disagreement scores after the RP80 method. In contrast, in the MovieLens dataset, SDAA performs better in group disagreement than Average+. At the same time, for both datasets, the group satisfaction scores for both methods are comparable, with SDAA clearly outperforming Average+ in the early iterations. This discrepancy in the

Table 6 NDCG and DFH scores for the ephemeral groups

	NDCG						DFH					
	Avg	SDAA	RP80	Par	SIAA	Avg+	Avg	SDAA	RP80	Par	SIAA	Avg+
MovieLens Dataset / Ephemeral Groups												
4+1	0.059	**0.062**	0.033	0.059	*0.060*	0.043	0.972	0.951	0.927	*0.976*	0.967	**0.981**
3+2	0.055	**0.058**	0.032	0.055	*0.056*	0.039	0.980	0.957	0.943	*0.985*	0.975	**0.986**
5 Dis	0.052	**0.055**	0.030	0.052	*0.053*	0.038	0.977	0.956	0.926	*0.983*	0.976	**0.986**
GoodReads Dataset / Ephemeral Groups												
4+1	0.069	**0.080**	0.025	*0.070*	*0.070*	0.053	0.898	0.828	0.769	*0.908*	0.895	**0.928**
3+2	0.066	**0.078**	0.023	0.066	*0.067*	0.050	0.846	0.782	0.681	*0.854*	0.842	**0.875**
5 Dis	0.063	**0.072**	0.025	0.047	*0.065*	0.050	0.907	0.842	0.820	*0.915*	0.902	**0.939**

With bold is the best value and with italics the second best

group disagreement values' performance allows Average+ to be constantly better than the rest of the methods in the GoodReads dataset. The discrepancy of performance between the two datasets is because the GoodReads dataset is more sparse than the MovieLens dataset. This negatively affects the single recommender system and reduces the number of items that are equally good for all members. Subsequently, SDAA tends to use the Least Misery approach, while Average+ considers the entire group recommendation list and overcomes the sparse obstacle.

In more detail, we can see from Figs. 1a, c, 2a, c, 3a and c that depict the stables groups, that Average and Par offer best group satisfaction scores but the worst group disagreement scores. The high group disagreement scores culminate in the Average's low overall performance. The Par method suffers from the same problem as Average. This is understandable since the Par method focuses more on the group satisfaction rather than disagreement. In the opposite case, the best disagreement scores are achieved by RP80, which mainly focuses on finding items with low disagreement, but at the same time, it also produces the worst group satisfaction scores. SIAA achieves the same group satisfaction as the Average but also produce lower group disagreement scores. Thus, out-performing Average.

For Average+, we can observe lower group satisfaction scores. This is expected since we build the group recommendation list based on an expanded list provided by Average aggregation. We do not limit ourselves on the items that are going to provide the best group satisfaction score (those are the top *k* returned by Average) but also consider items that may offer lower satisfaction but considerable better group disagreement scores. Consequently, the satisfaction scores for Average+ are lower than the scores of the Average method.

8.2.2 Ephemeral groups

As we can observe in the F-score tables, the aggregation methods' performance vastly differs in the ephemeral groups' scenario. Overall, Average+ and SIAA are the best for all cases, with the difference between them to be around 8% in the early iterations and dropping to 1% in the latter ones.

We can observe a discrepancy on the performance of the SDAA method. The constant change of members in the group has an adverse effect on the SDAA method, with its performance dropping drastically. The addition of a new unknown member at each iteration forces the SDAA to utilize higher α values. The new member has no previous satisfaction score, which in turn makes the value of α to be the highest possible at each iteration (9). SDAA does not perform well with high α values (Stratigi et al., 2020). With new members repeatably being added to the group, SDAA cannot keep an optimal performance and is outperformed by the classic Average method.

In the early iterations, Par has comparable results as SIAA and Average+, but in latter iterations the difference becomes greater. This is because in the latter iterations, with a high probability the most relevant items have already been suggested to the group (and cannot be suggested again, per our experimental scenario), so the old members are disadvantaged. Their items were suggested in the previous iterations and the interests of the new members are more likely to be recommended. Additionally, the Par method only considers the current iteration, without taking into account if a member was not satisfied in any of the previous iterations. This culminates to the high disagreement scores, observed in the ephemeral group scenario.

In Figs. 1b, d, 2b, d, 3b and d, we examine the performance of the methods in more detail. As in the case with the stable groups, the best group satisfaction scores are achieved by the

Average and Par methods and the best group disagreement scores by Average+. In contrast to the stable groups scenario, Average+ now achieves better disagreement scores than RP80.

Although in the early iterations, Average+ has lower satisfaction scores than the rest of the methods except RP80, in the latter ones, it manages to outperform SDAA. However, it is considerably below both Average, Par and SIAA. SIAA has maintained exemplary group satisfaction scores throughout all the scenarios, but with ephemeral groups, it can now achieve good disagreement scores. This is due to the user-focused concept of SIAA. It can cope better with the added new group member than a group-focused method like SDAA.

8.2.3 Stable versus ephemeral groups

In general, all aggregation methods perform better overall for ephemeral groups rather than the stable ones. This can be seen in the latter iterations when comparing the ephemeral and the stable groups scenarios. This is expected, since our group recommender system does not suggest items to the group that it has already suggested before. In the stable groups, the best items for each member were probably suggested in the early iterations. Now that the members interchange, the system is able to suggest to the new members items that are the best for them.

This can be further corroborated by the satisfaction scores shown in the figures. For the ephemeral groups, we can observe that the group satisfaction scores tend to remain high throughout the iterations, without the notable slope noted in the stable groups. Furthermore, in correlation to the high satisfaction scores, the disagreement scores are again lower for the ephemeral groups than the stable ones. The new interests that the added members represent in the sequential group recommendation help the system maintain high performance throughout many iterations.

We can also observe that the stable groups' disagreement scores tend to increase uniformly as the satisfaction scores drop. This again represents that the group recommender system achieves increasingly lower quality suggestions for the group over the passage of the iterations. Meanwhile, for the ephemeral groups, we can see high and low spikes in the disagreement scores. These spikes represent the new group members and how their inclusion slows down the overall disagreement scores' deterioration.

8.2.4 Methods quality

In Tables 5 and 6, we show the NDCG and DFH scores for each aggregation method, for the various test conditions. For both stable and ephemeral groups and both datasets, SDAA is the best performing, as far as NDCG scores, while SIAA is the second best. Average+ has low NDCG scores since we build the group recommendation list with items that can have lower relevance to the group. It is able to report back items that are relevant (it generates good satisfaction scores), but most of them are not the best for each user; thus, it has lower NDCG scores. Once again, we observe a difference in the quality of the methods when they are applied to stable and ephemeral groups. These results supplement the results from before, where the methods perform better in the ephemeral group scenario.

In contrast to NDCG scores, Average+ has the best scores for DFH. This means that Average+ succeeds in proposing at least one item in the group recommendation highly relevant to each member. However, the rest of the list is composed of unsatisfactory items (in terms of relevance), hence the low NDCG scores. On the other hand, the DFH scores for SDAA are relatively low. This indicates that SDAA, although able to propose many items relevant to the members, high NDCG scores, cannot suggest the best items for these users.

Table 7 The percentage of time each scenario was enacted for all 100 groups

	MovieLens	GoodReads
Nothing	36.58%	35.45%
Insert	34.99%	35.98%
Remove	28.43%	28.57%

8.2.5 Fluctuating group size

In this final test case, we want to observe the performance of each aggregation method, when the size of the group is not fixed. We examine the 4+1 group format. We want to keep this group format for the duration of the experiment i.e., for all 15 iterations. Since the group size is not fixed, we require that at each iteration 20% of the group members be dissimilar to the rest. For example, when the group size is 7, then 6 members should be similar to each other and 1 is dissimilar to them (20% of 7 is 1.4, which gives us 1 dissimilar user when rounding down). If in the next iteration we add another user to the group, then that user will have to be dissimilar to the 6 others (20% of 8 is 1.6, which gives us 2 dissimilar users when rounding up).

With equal probability we enact one of the three scenarios:

– **Nothing:** No new members will enter or exit the group.
– **Insert:** Insert a new member to the group.
– **Remove:** Remove a member from the group.

In Table 7, we show the number of times each scenario was enacted for all 100 groups in the two datasets. In Fig. 4, we show the group satisfaction and disagreement scores for the 15 iterations, and in Table 8 the average values after 15 iterations for the F-Score, NDCG and DFH.

We can observe the same trend as the previous results in Fig. 1b and d. Meaning that the Average and Par methods offer the best group satisfaction scores and simultaneously the worst group disagreement scores. SIAA performs comparable good in satisfaction to Average and Par but has better group disagreement scores and finally Average+ has noticeable lower satisfaction scores but far better disagreement scores.

This is further corroborated by the values in Table 8, where both SIAA and Average+ have the best F-Score. The high disagreement scores that Par produces lower its overall

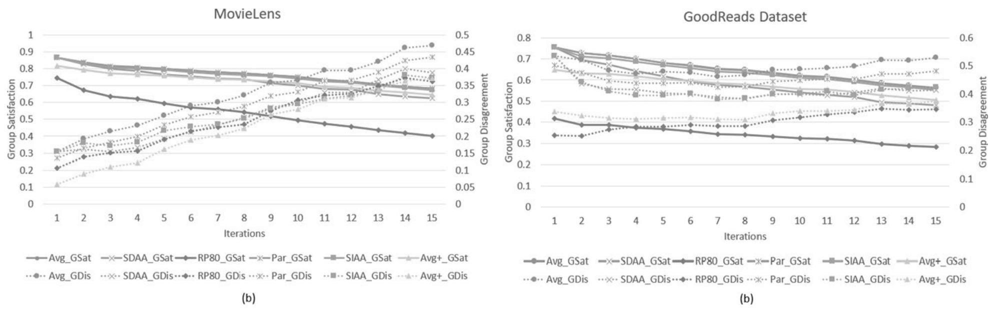


Fig. 4 Group Satisfaction and Disagreement for all aggregation methods for 15 iterations for fluctuated group size, using MovieLens and GoodReads

Table 8 Average scores after 15 iterations

	MovieLens			GoodReads		
	F-Score	NDCG	DFH	F-Score	NDCG	DFH
Avg	0.719	*0.052*	0.963	0.567	0.062	0.915
SDAA	0.734	**0.055**	0.920	0.579	**0.069**	0.827
RP80	0.629	0.025	0.801	0.459	0.020	0.710
Par	*0.736*	*0.052*	*0.975*	0.591	*0.064*	*0.929*
SIAA	**0.749**	*0.052*	0.963	*0.608*	0.063	0.914
Avg+	**0.749**	0.039	**1.000**	**0.618**	0.050	**0.969**

With bold is the best value and with italics the second best

performance. SDAA identifies more relevant items to the group in terms of quantity, as indicated by the highest NDCG score, but under-performs in terms of quality with low DFH scores.

The effect of the fluctuating group size can be observed in the lower group satisfaction scores and the higher group disagreement scores. If we again compare these results to Fig. 1b and d, we can see that all methods perform slightly worse when the size of the group is not fixed. This is an expected result. As it has been shown in the literature, the size of the group has an adverse effect in the performance of a group recommender system. The more members a group has the more difficult is for the system to find items that are relevant to all members. In Table 7, we show that in most cases we add a member to the group, which results in groups bigger than 5; the size used in the previous experiments.

9 Related work

Group recommendations One of the most popular methodology to achieve group recommendations is to apply a standard recommendation algorithm to each group member individually and aggregate their lists into one. During the aggregation phase, many criteria can be taken into account. Yuan et al. (2014) proposes a group recommendation model that considers the influence that each member has on the group's final choice, stating that a member has more influence on the group if he/she is more knowledgeable about the recommended items. In our work, we propose that if a group member is more dissatisfied than the rest, then that member will have more influence in the group decision. Cao et al. (2018) learns the aggregation strategy from data, which is based on the recent developments of attention network and neural collaborative filtering (NCF), while we dynamically change our proposed aggregation method based on the satisfaction of the group members. Yin et al. (2019) also utilizes an attention mechanism as well as a Bipartite Graph Embedding Model (BGEM) to learn the influence that each member has on the group decision. Salehi-Abari and Boutilier (2015) exploits the social interactions between the group members, via a preference-oriented social network, in order to make group decisions without complete knowledge of the group members' preferences. In our work, we require the full knowledge of the group members' preferences. Vinh Tran et al. (2019) learns the aggregation strategy based on the interactions between group members. To simulate how a group consensus is reached, they model these interactions as multiple voting processes, and proposed a stacked social self-attention network to learn the voting scheme of group members. Qin et al. (2018)

offers a novel approach to producing recommendations for a large group of people by dividing the big group into different interest subgroups. For each subgroup, they find a potential candidate set of media-user pairs and finally aggregate the CF produced recommendation lists for each pair.

Recently, there is also some research work on ephemeral groups. Quintarelli et al. (2016) determines the preference of a group that meets for the first time by combining the group members' individual preferences based on their contextual influence, where the contextual influence represents the ability of a member to guide the decisions of the group. Liu et al. (2012) proposes a probabilistic model, namely the personal impact topic (PIT) model. An individual member's preference is modeled as a distinct distribution over the system's topics. Each topic is expressed as a distinct distribution over all the items known to the system. We explicitly use the users' profiles (ratings) to provide the group with a group list.

Fairness in group recommendations Some of the first approaches to achieving fairness Pitoura et al. (2021, 2020) in group recommendations, exploit ideas from voting theory. For example, Naamani Dery et al. (2010) assumes that the recommender has probabilistic knowledge about the distribution of users' ratings, and through voting theory recommends to the group a "winning" item. From a different perspective, Guzzi et al. (2011) proposes a group recommendation method by allowing a group member to comment on the choices of the rest of the group. This allows each user to get new recommendations similar to the proposals made by other group members and to communicate the rationale behind their own counter-proposals.

More recently, Serbos et al. (2017) proposes two definition of fairness: *fairness proportionality* and *envy-freeness*. In the former, the user u considers the list of recommended items fair for him/her, if there are at least m items that the user likes. In the latter, u considers the package fair, if there are at least m items for which the user does not feel envious. Xiao et al. (2017) presents yet another definition of fairness. It defines a utility score for each group member based on how relevant are the recommended items to them. Then, it models fairness as a proximity of how balanced the utilities of users are when group recommendations are given. Sacharidis (2019) defines a user's utility to be the similarity between the user's individual and the group recommendation lists. They construct the group recommendation list by considering sets of N-level Pareto optimal items. Kaya et al. (2020) proposes the notion of rank-sensitive balance in order to achieve fairness during the aggregation phase. All these works, for achieving fairness, consider one instance of group recommendations and do not take into account the sequential group recommendation problem, as we do in our work.

Sequential recommendations In general, there are three categories of sequential recommenders, and they are divided based on how many past user interactions they consider: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* (Quadrana et al., 2018). In the first approach, only the last N user actions are considered (Cheng et al., 2013; Lian et al., 2013; Liu et al., 2016). This is because the system has logged a huge amount of historical data for the user, with many of them be duplicates, which do not offer relevant information to the system. In session-based recommendations, only the last interaction of the user with the system is used. They are typically found in news (Garcin et al., 2013) and advertisement (Hidasi et al., 2016) recommenders. In the last category, we have information about both the last interaction of the user with the system, as well as the history of the user. These recommenders are often implemented in e-commerce or for app suggestions (Hariri et al., 2012; Jannach et al., 2015; Quadrana et al., 2017). In our work, we use the last method to approach sequential group recommendations.

Hansen et al. (2020) proposes another session-aware system for music recommendations. It uses a neural network architecture that models users' preferences as a sequence of embeddings, one for each session, suggesting that the user's recent selections and the session-level contextual variables (such as time and device used) are enough to predict the tracks a user will listen to. Borges and Stefanidis (2019) presents a multi-round recommender system using Variational Autoencoders (VAEs) and tries to achieve fairness during multiple rounds by introducing randomness in the regular operation of VAE, while (Borges & Stefanidis, 2021; 2020) penalize scores given to items according to historical popularity for mitigating the bias and promoting diversity in the results. The above works have been done for single user recommenders, and to our knowledge, our work is the first one in sequential group recommendations that handles the notion of fairness.

This article extends (Stratigi et al., 2020), where we first presented SDAA. We now examine two more aggregation methods, SIAA and Average+, designed to deal with sequential group recommendations. Instead of focusing only on the group, like with SDAA, now with SIAA, we focus more on the user side. We also examine a heuristic method, Average+, where we try to find the best possible items to offer to the group based on the satisfaction and disagreement score they generate. In addition, we also consider the ephemeral group scenario, where at the end of each iteration, a group member leaves the group, and a new one enters the group.

10 Conclusions

In this paper, we introduce the notion of sequential group recommendations. We treat the single user recommender system as a black-box, and focus on aggregating the individual group members' recommendation lists into a group list. We propose three new methods for aggregating group members' recommendation lists. Specifically, SDAA focuses on the group as a whole, and dynamically, based on the degree of users' satisfaction, balances the advantages of the average and least misery methods. With SIAA, we concentrate on singular members, focus on their needs, and assign personal weights. Finally, we propose the Average+ method that takes advantage of the satisfaction scores that a classic average aggregation can produce and mitigates the high disagreement scores by incrementally building the group list with the best possible item. These aggregation methods focus on the sequential aspect of the group recommendation process and how we can achieve fairness for all group members in multiple recommendation rounds. We evaluate the proposed methods using two large real datasets, MovieLens and GoodReads, and test the methods for three different group types for both stable and ephemeral groups. We experimentally show the effectiveness of our methods. SDAA outperforms in the latter iterations for the stable group scenario. However, in the ephemeral groups, SDAA cannot perform optimally primarily due to the users' transitory attitude. In contrast, SIAA and Average+ are the best for the ephemeral groups and among the best in the stable groups' scenario.

Funding Open Access funding provided by Tampere University including Tampere University Hospital, Tampere University of Applied Sciences (TUNI).

References

Amer-Yahia, S., Roy, S. B., Chawlat, A., Das, G., & Yu, C. (2009). Group recommendation: Semantics and efficiency. *PVLDB*, *2*(1), 754–765.

Baltrunas, L., Makcinskas, T., & Ricci, F. (2010). Group recommendations with rank aggregation and collaborative filtering. In *RecSys*.

Borges, R., & Stefanidis, K. (2019). Enhancing long term fairness in recommendations with variational autoencoders. In *MEDES*.

Borges, R., & Stefanidis, K. (2020). On measuring popularity bias in collaborative filtering data. In *Proceedings of the workshops of the EDBT/ICDT 2020 joint conference, Copenhagen, Denmark, March 30, 2020*, Vol. 2578.

Borges, R., & Stefanidis, K. (2021). On mitigating popularity bias in recommendations via variational autoencoders. In *SAC '21: The 36th ACM/SIGAPP symposium on applied computing, virtual event, Republic of Korea, March 22-26, 2021* (pp. 1383–1389). ACM.

Cao, D., He, X., Miao, L., An, Y., Yang, C., & Hong, R. (2018). Attentive group recommendation. In *SIGIR*.

Cheng, C., Yang, H., Lyu, M. R., & King, I. (2013). Where you like to go next: Successive point-of-interest recommendation. In *International joint conference on artificial intelligence*.

Garcin, F., Dimitrakakis, C., & Faltings, B. (2013). Personalized news recommendation with context trees. arXiv:1303.0665.

Guzzi, F., Ricci, F., & Burke, R. (2011). Interactive multi-party critiquing for group recommendation. In *RecSys*.

Hansen, C., Hansen, C., Maystre, L., Mehrotra, R., Brost, B., Tomasi, F., & Lalmas, M. (2020). Contextual and sequential user embeddings for large-scale music recommendation. In *RecSys*.

Hariri, N., Mobasher, B., & Burke, R. (2012). Context-aware music recommendation based on latenttopic sequential patterns. In *RecSys*.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions Interaction Intelligence Systems*, *5*(4), 19:1–19:19.

Hidasi, B., Quadrana, M., Karatzoglou, A., & Tikk, D. (2016). Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*.

Jannach, D., Lerche, L., & Jugovac, M. (2015). Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys*.

Kaya, M., Bridge, D., & Tintarev, N. (2020). Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In *RecSys*.

Lian, D., Zheng, V. W., & Xie, X. (2013). Collaborative filtering meets next check-in location prediction. In *WWW*.

Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI conference on artificial intelligence*.

Liu, X., Tian, Y., Ye, M., & Lee, W.C. (2012). Exploring personal impact for group recommendation. In *RecSys*.

Machado, L., & Stefanidis, K. (2019). Fair team recommendations for multidisciplinary projects. In *WI*.

Naamani Dery, L., Kalech, M., Rokach, L., & Shapira, B. (2010). Iterative voting under uncertainty for group recommender systems. In *RecSys*.

Ntoutsi, E., Stefanidis, K., Nørvåg, K., & Kriegel, H. (2012). Fast group recommendations by applying user clustering. In *ER*.

Ntoutsi, E., Stefanidis, K., Rausch, K., & Kriegel, H. (2014). "strength lies in differences": Diversifying friends for recommendations through subspace clustering. In *CIKM*.

Pitoura, E., Koutrika, G., & Stefanidis, K. (2020). Fairness in rankings and recommenders. In *Proceedings of the 23rd international conference on extending database technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020* (pp. 651–654).

Pitoura, E., Stefanidis, K., & Koutrika, G. (2021). Fairness in rankings and recommendations: An overview. arXiv:2104.05994.

Qin, D., Zhou, X., Chen, L., Huang, G., & Zhang, Y. (2018). Dynamic connection-based social group recommendation. IEEE TKDE.

Quadrana, M., Cremonesi, P., & Jannach, D. (2018). Sequence-aware recommender systems. *ACM Computer Survey*, *51*(4), 66:1–66:36.

Quadrana, M., Karatzoglou, A., Hidasi, B., & Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*.

Quintarelli, E., Rabosio, E., & Tanca, L. (2016). Recommending new items to ephemeral groups using contextual user influence. In *RecSys*.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*.

Sacharidis, D. (2019). *Top-n group recommendations with fairness. SAC '19*. New York, NY USA: Association for Computing Machinery. https://doi.org/10.1145/3297280.3297442.

Salehi-Abari, A., & Boutilier, C. (2015). *Preference-oriented social networks: Group recommendation and inference. RecSys '15*. New York, NY USA: Association for Computing Machinery. https://doi.org/10.1145/2792838.2800190.

Serbos, D., Qi, S., Mamoulis, N., Pitoura, E., & Tsaparas, P. (2017). Fairness in package-to-group recommendations. In *WWW*.

Stratigi, M., Kondylakis, H., & Stefanidis, K. (2017). Fairness in group recommendations in the health domain. In *ICDE*.

Stratigi, M., Kondylakis, H., & Stefanidis, K. (2018). Fairgrecs: Fair group recommendations by exploiting personal health information. In *DEXA*.

Stratigi, M., Nummenmaa, J., Pitoura, E., & Stefanidis, K. (2020). Fair sequential group recommendations. In *ACM SAC*.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence vol 2009.

Vinh Tran, L., Nguyen Pham, T. A., Tay, Y., Liu, Y., Cong, G., & Li, X. (2019). *Interact and decide: Medley of sub-attention networks for effective group recommendation. SIGIR'19*. New York, NY USA: Association for Computing Machinery. https://doi.org/10.1145/3331184.3331251.

Wan, M., Misra, R., Nakashole, N., & McAuley, J. (2019). Fine-grained spoiler detection from large-scale review corpora.

Xiao, L., Min, Z., Yongfeng, Z., Zhaoquan, G., Yiqun, L., & Shaoping, M. (2017). Fairness-aware group recommendation with pareto-efficiency. In *RecSys*.

Yin, H., Wang, Q., Zheng, K., Li, Z., Yang, J., & Zhou, X. (2019). Social influence-based group representation learning for group recommendation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (pp. 566–577). https://doi.org/10.1109/ICDE.2019.00057.

Yuan, Q., Cong, G., & Lin, C.Y. (2014). Com: A generative model for group recommendation. In *KDD*.

PUBLICATION III

SQUIRREL: A framework for sequential group recommendations through reinforcement learning

Maria Stratigi, Evaggelia Pitoura, and Kostas Stefanidis

Inf. Syst. 112 (2023), p. 102128
DOI: 10.1016/j.is.2022.102128. URL: https://doi.org/10.1016/j.is.2022.102128

Information Systems 112 (2023) 102128

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

SQUIRREL: A framework for sequential group recommendations through reinforcement learning

Maria Stratigi [a,*], Evaggelia Pitoura [b], Kostas Stefanidis [a]

[a] Tampere University, Tampere, Finland
[b] University of Ioannina, Ioannina, Greece

A R T I C L E   I N F O

Article history:
Received 1 February 2022
Received in revised form 20 August 2022
Accepted 13 September 2022
Available online 23 September 2022
Recommended by Dennis Shasha

MSC:
0000
1111

Keywords:
Multi-round recommendations
Group recommendations
Rank aggregations

A B S T R A C T

Nowadays, sequential recommendations are becoming more prevalent. A user expects the system to remember past interactions and not conduct each recommendation round as a stand-alone process. Additionally, group recommendation systems are more prominent since more and more people are able to form groups for activities. Subsequently, the data that a group recommendation system needs to consider becomes more complicated — historical data and feedback for each user, the items recommended and ultimately selected to and by the group, etc. This makes the selection of a group recommendation algorithm to be even more complex. In this work, we propose the SQUIRREL framework — SeQUentIal Recommendations with ReinforcEment Learning, a model that relies on reinforcement learning techniques to select the most appropriate group recommendation algorithm based on the current state of the group. At each round of recommendations, we calculate the satisfaction of each group member, how relevant each item in the group recommendation list is for each user, and based on this the model selects an action, that is, a recommendation algorithm out of a predefined set that will produce the maximum reward. We present a sample of methods that can be used; however, the model is able to be further configured with additional actions, different definitions of rewards or states. We perform experiments on three real world datasets, 20M MovieLens, GoodReads and Amazon, and show that SQUIRREL is able to outperform all the individual recommendation methods used in the action set, by correctly identifying the recommendation algorithm that maximizes the reward function utilized.

1. Introduction

Recommendation systems have developed into one of the most influential part of research for projects, spanning from e-health to movie recommendations. This is because recommendation systems are to a great degree responsible for the end-users satisfaction with an application. They analyze a user's past interactions (in the various forms these interactions may take like reviews, like/dislike, clicks, etc.) to enhance the experience of the user. To achieve this, the recommendation systems have to consider not only the user's current information but also their past interactions with the system. That is, the system should not only rely on the current session but also consider historical ones and the response that the user had on them, i.e., which items the user ultimately picked or liked. The system should consider a historical *sequence of interactions* or *rounds of recommendations* for a user, instead of focusing on just the current session. This is a relatively new approach to recommendations, since typically most systems examine the user's available ratings but do not consider how well the user received the past recommended items.

In this work, we consider a sequential recommendation system as a system that considers a sequence of rounds of recommendations for users. A previous work done on sequential recommendations is [1], where user and item dynamics are captured by historical sequences on the user side and the item side, respectively. In [2], a neural network is used, and every round of recommendations is represented by an embedding of the users' preferences. [3] uses variational autoencoders (VAEs) to propose a multi-round recommendation system, which involves introducing randomness into the regular operation of VAEs in order to promote fairness, while [4] penalizes items for their historical popularity in an effort to minimize bias and promote diversity.

In addition to sequential recommendation, in recent years, greater attention is given to group recommendations, where the recommendation system needs to balance the interests of a group of users instead of focusing on just one user. For example, assume a group of friends that wants to watch a movie. The group recommendation process is more complicated than a single recommendation system, since each group member has their own likes and dislikes and the group recommendation system needs

* Corresponding author.
E-mail addresses: maria.stratigi@tuni.fi (M. Stratigi), pitoura@cs.uoi.gr (E. Pitoura), konstantinos.stefanidis@tuni.fi (K. Stefanidis).

https://doi.org/10.1016/j.is.2022.102128
0306-4379/© 2022 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

to balance them, in order to propose a group recommendation list that ideally is relevant to all members interests.

There are many different ways to achieve group recommendations, but one of the most popular is to employ a single recommendation system for each group member and receive their corresponding recommendation lists. Then, the group recommendation system takes over and tries to combine these lists into one group recommendation list via an aggregation method [5]. Many strategies, with different advantages and disadvantages, can be applied during this aggregation step. However, even after years of research, and according to the Arrow's Theorem [6], no aggregation function has been proven to be the most effective one.

Both of these areas of recommendation systems, sequential and group recommendations, are complex on their own right, but they become even more complex when they are combined. This new area of recommendations, the *sequential group recommendations*, is important to explore in more depth. Similarly to single recommendation systems, a sequential group recommender will be able to provide to a group a more robust experience if it analyzes its past interactions with that group. Specifically, a sequential group recommender has to resolve the challenges of group and sequential recommendations simultaneously. That is, it has to balance the interest of all group members and propose relevant items to all, while accurately determining how to infer users' dynamic preferences between rounds of recommendation.

Although there are many ways one can approach group recommendations, it is not always clear which approach is the best for each test case. The vast amount of group recommendation methods and their widely distinct approaches to recommendations, make it very difficult and time consuming to not only evaluate all of them but to also see which one works best for each test scenario. Some of the methods work very well in a specific recommendation domain, for example movie recommendations, but straggle when they are applied to a different one [7]. As a result it is not an easy task to simply transfer a recommender system to another domain.

In this work, we propose a model that is able to be applied to any given domain without major alterations and minimum time invested, while simultaneously, it is also able to maintain high performance. The SQUIRREL framework, **S**e**QU**ent**I**al Group **R**ecommendations through **R**einforc**E**ment **L**earning, is a model that is based on Reinforcement Learning (RL), a natural choice, since the sequential nature of RL directly mirrors the sequential nature of our problem. It consists of three main components: state, actions and reward. The state describes the current status of the group, the actions are the different group recommendation methods that can be applied and finally, the reward is the primary goal that the system wants to achieve. In most recommendation systems that goal is the group members' satisfaction with the proposed items, meaning how relevant are the items in the group recommendation list for each group member.

At each round of recommendations, the system examines the state of the group, for example, how satisfied each group member is, and based on that, it selects an appropriate action to make, i.e., it selects a group recommendation method to apply. Based on this selection, a group recommendation list is produced and returned to the group and a reward is calculated. Then, the state of the group is updated to reflect the changes made by the action, i.e., how satisfied are the group members after the latest recommendation round. Note that to simulate time passing between each round of recommendations, we augment the system with additional data (i.e., the users rate more items, hence the user profiles change between rounds of recommendations).

All the components of the model can be configured to the specifications of the user and the criteria of the test scenario, and are easily altered. For instance, if we want to evaluate a new group recommendation method, then that method is simply added as a new action in the model. If the goal of the recommender needs to change, for example, the system's new primary goal is changed from promoting satisfaction to promoting diversity, then a new reward can be defined. If the system is transferred to a new recommendation domain then a new state can be specified. However, it is worth noting that if a component changes, then the model will have to be trained again.

Since the variations of scenarios and group recommendation methods that can be applied are numerous, in this work we mainly focus on rank aggregation group recommendation methods. As the state of our model, we use the notion of user's satisfaction, i.e., how relevant the items in the group recommendation list are to each group member. Finally, we study two reward functions: one based on satisfaction and the second based on the combination between user's satisfaction and user's disagreement, which is defined as the difference between the most and least satisfied group member.

Overall, the main contributions of our work are the following:

- We introduce the SQUIRREL framework, a model that based on principles of reinforcement learning is able to identify the best possible group recommendation method to apply based on the information it has available and the goal it wants to fulfill.
- We focus on rank aggregation group recommendation methods, and exploit the notion of user satisfaction and user disagreement to define the various components of our model.
- We evaluate the SQUIRREL model and all individual methods used as actions, using three real world datasets, 20M MovieLens, GoodReads and Amazon. Additionally, we examined the behavior of all methods on different types of groups.

The rest of the paper is structured as follows. In Section 2, we formally present the SQUIRREL model, while in Section 3, we go into more detail about all the components of the model; state, actions and rewards. In Section 4, we showcase the datasets we utilized, and in Section 5, we present the experimental process and analyze the results. Furthermore, in Section 6, we describe the related work, and finally, in Section 7, we conclude this work.

2. Reinforcement learning for group recommendations

Let I denote a set of data items and U a set of users. G denotes a group of users where $G \subseteq U$. For each user u in the group, we denote as B_u^j an ordered list of recommended items, as they have been generated by a single recommender system at a specific recommendation round j for user u. At round j, the SQUIRREL model chooses an appropriate aggregation function based on the current state of the group, to combine the individual members' recommendation lists B_u^j into one group recommendation list GL_G^j.

We apply the SQUIRREL model for a sequence of rounds in order to produce μ group recommendation lists for a group G, defined as $\mathcal{GR} = (GL_G^1, GL_G^2, \ldots, GL_G^{\mu})$. At each round, we calculate each group members' individual *user utility* scores, as well as the *group utility* scores. The first set of scores portrait how satisfied each member is with the proposed group recommendation list, as well as the disagreement between the group members, i.e., the variance between the satisfaction scores of the group members. The latter set of utility scores describe the degree of satisfaction and disagreement for the entire group. Namely, how can we exploit the user utility scores to reflect the entire group. We further expand on these notions in Section 3.2.

The SQUIRREL model can be described as a Markov decision process, where an agent interacts with an environment E, in order

Table 1
Summary of the notations used in this work.

Notation	Definition	Notation	Definition
I	Set of items	GL_G^j	Group recommendation list at round j
U	Set of users	$\mathcal{GR}$	Sequence of group recommendations
G	Group	μ	Number of rounds in the sequence
u	User	SQUIRREL model	
d	Item	S	State of the environment
j	Round	A	Set of actions
B_u^j	Recommendations list for user u at round j	$P_a(s, s')$	Probability of transitioning from state s to state s'
$B_{u,k}^j$	Top k recommendations for user u at round j	$R_a(s, s')$	Reward from transitioning from s to s' under action a
$p_j(u, d)$	User u's relevance score for item d at round j	π	Policy of the model

to maximize the accumulative reward after each recommendation round. The Markov decision process can be described by a tuple of (S, A, P, R), where:

- ***S*** is a continuous space that describes the environmental state, i.e., the group state and we express it via the utility scores of the group members. We keep an individual state for each group member u at each round j. It can be defined as $s_u^j = satO(u, j)$, where $satO$ is the overall satisfaction of user u at round j. This satisfaction score will be discussed in detail in Section 3.1. Briefly, it describes how relevant are the data items in the group recommendation list, compared to the best case scenario for the user, which is their individual recommendations B_u^j.
- A is a set of distinct actions that consists of the different aggregation functions employed by the SQUIRREL model, where $|\text{A}| = m$. These functions can range from simplistic ones, like a classic Average, to far more complex ones like SDAA first proposed in [8], without any restriction in regards to the number of actions we can include. We go in more detail about each aggregation method used in Section 3.3.
- $P_a(s, s')$ defines the probability to transition from state s to state s' during round j under the action a. Formally, $P_a(s, s') = Pr(s_{j+1} = s' | s_j = s, a_j = a)$.
- $R_a(s, s')$ is the reward gained from transitioning from state s to state s'. The reward describes the quality of recommendations given by the model. We define two reward functions by utilizing group utility scores. First, we examine only the overall satisfaction of the group by averaging the individual satisfaction of all group members. Second, in addition to the overall satisfaction, we also consider the disagreement between the group members. This is defined as the difference in the satisfaction scores between the most and least satisfied group member. We expand on these reward functions further in Section 3.2.

The goal of the model is to find a policy $\pi(a|s)$ that takes action $a \in A$ during state $s \in S$ in order to maximize the expected discounted cumulative reward after μ recommendation rounds:

$$max\, \mathbb{E}[R(\mu)] \tag{1}$$

where

$$R(\mu) = \sum_{t=0}^{\mu} \gamma R_{\alpha}(s, s') \tag{2}$$

with $0 \leq \gamma \leq 1$.

It is worth noting that the individual component of the SQUIRREL model can be altered and fine-tuned for a specific purpose. For example, an application that wants to minimize the differences between users' perceived overall performance may need to define a different state and/or reward, while an application that wants to find the best variable for an aggregation function may want to define a different action space. For instance, in the case of a simple Weight Sum aggregation function [9], the action space will be the different weights.

As any RL model, SQUIRREL requires a training phase. To achieve this, the system depends upon having a large amount of data to utilize for this training phase. During training, the model will learn what action (i.e., what aggregation method) is more effective to use based on the current state of the environment. If any of the model's components (state, actions or reward) is changed, then the model will have to be re-trained.

In Fig. 1, we show how a recommendation round is structured in the SQUIRREL model. At the beginning of the round j, the group is given to a single user recommender system that produces a set of recommendation lists for each group member, B_u^j. These lists are then given to the SQUIRREL model, where the agent observes the state of the environment S_j, mainly how satisfied the current group is. Then it selects an appropriate action α_j to aggregate the lists B_u^j. This results in the transitioning of the model to the next state S_{j+1} where we update the overall satisfaction of the users and the calculated reward R_{j+1}. Finally, the model returns to the group the generated group recommendation list GL_G^j. For ease of readability, Table 1 describes all the notations used in this work.

3. The SQUIRREL model

In this section, we go into details for every major component of the SQUIRREL model, namely state, action and reward. These components are flexible and can be adjusted to the needs of the user. For this work, to demonstrate our model we utilized elements from previous works in group recommender systems [8,10,11]. Namely, to define the state of the model, we use *user utility scores* that describe the satisfaction of each user individually with the group recommendations. For the reward, we focus more on the group as an entity with *group utilities* scores which showcase how relevant are the recommended data items for the entire group. Finally, we utilize a number of different aggregation functions as the actions of the model.

3.1. State definition via users utility scores

One of the most essential parts of the SQUIRREL model, is the definition of the state of the environment. The state is the component that describes the current status of the group members. Accordingly, we need a state that is focused on each group member individually. This will help the model to make the best possible choice in the policy, without accidentally disregard any group member, something that is probable if we only consider the group in its entirety. It is often the case where a more generalized picture, will disguise individual needs.

In order to describe how satisfied a user is with the group recommendation list, we calculate two different utility scores. Their *satisfaction* with the proposed items and their *disagreement* with the rest of the group. As in a typical group recommender, we apply a recommendation algorithm for each group member u

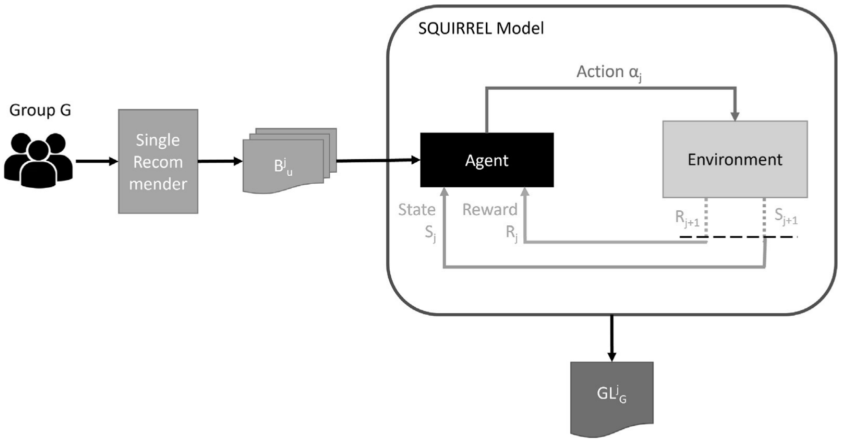


Fig. 1. The SQUIRREL Model.

and generate their corresponding individual recommendation list B^j_u at round j. We consider the user's individual recommendation list, as their ground truth, making SQUIRREL more flexible, by allowing it to consider the single user recommendation system as black-box. Note that since after a recommendation round we augment the system with additional ratings, the ground truth of each user changes between rounds.

Let $B^j_{u,k}$ be a list that contains the k items with the highest scores in the B^j_u, and GL^j_G be the group recommendation list that also contains k items. To define a user's satisfaction with the items in the group recommendation list GL^j_G, we will directly compare them to the best items for the user u, i.e., the $B^j_{u,k}$ list. This is commonly referred as the individual loss with respect to the group recommendations [12]. Specifically:

$$sat(u, GL^j_G) = \frac{\sum_{d \in GL^j_G} p_j(u, d)}{\sum_{d \in B^j_{u,k}} p_j(u, d)}. \tag{3}$$

where $p_j(u, d)$ defines the score of the data item d as it appears in the list B^j_u. That is, the recommendation score of item d for user u at round j, as it has been calculated by a single user recommendation algorithm. In the nominator, we calculate the user's satisfaction based on the group recommendation list. For every item in GL^j_G, we sum the score as they appear in each user's top-k list. The denominator, calculates the ideal case for the user, by simply sum the scores of the k top items in the user's individual recommendation list. This way, we are able to normalize the user's satisfaction score. Note that we do not use the scores as they appear in the group list, but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process somewhat distorts the group members' individual opinions, we opt to take into consideration only the personal preference scores of each group member.

To demonstrate how satisfied a user is after a sequence of μ iterations, we define the overall satisfaction of a user. Meaning, that this score demonstrates how satisfied the user is with the performance of the system after multiple rounds of recommendations.

Definition 1 (*Overall Satisfaction*)**.** The overall satisfaction of user u with respect to a sequence $\mathcal{GR}$ of μ rounds of recommendations is the average of the satisfaction scores after each round:

$$satO(u, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u, GL^j_G)}{\mu}. \tag{4}$$

After each round, we update the internal state of the model to be the new overall user satisfaction. For each group member, we keep their overall satisfaction up to the current round.

3.2. Reward definition via group utility scores

The reward that is generated by an action given the state of the environment, is the only mechanism that the model has to determine if an action it took was an appropriate one. This reward can be very flexible in its definition based on what we want our model to achieve. Commonly, we want our model to propose items that are relevant to the entire group. For this, we need to define the *group utility scores*, that portrait how good are the recommendations for the group as a whole.

One option to define the reward is via the group satisfaction score. We generalize the user satisfaction score to describe the satisfaction of the entire group. This score will indicate how well the system is able to balance the individual needs of the group members. A high group satisfaction score means that the system is able to identify items that are relevant to the majority of the group members, and alternatively, a low group satisfaction score indicates that the system has failed in this task.

Specifically, we define the group G's satisfaction concerning a group recommendation list GL^j_G to be the average of the satisfaction of the users in the group:

$$groupSat(GL^j_G) = \frac{\sum_{u \in G} sat(u, GL^j_G)}{|G|}. \tag{5}$$

Subsequently, we define the overall group satisfaction of a group G for a recommendation sequence $\mathcal{GR}$ of μ group recommendations as:

$$groupSatO(\mathcal{GR}) = \frac{\sum_{u \in G} satO(u, \mathcal{GR})}{|G|}. \tag{6}$$

This overall group satisfaction can be used for expressing the reward achieved at recommendation round j by action a.

$$R_s(\mathcal{GR}^j) = groupSatO(\mathcal{GR}^j) \tag{7}$$

where $\mathcal{GR}^j$ refers to all the rounds up to the jth one.

However there is a drawback to the definition of the group satisfaction score, that is we consider the average of the group members' individual satisfaction scores. This can lead us to somehow ignore the dissatisfaction of a user. If all members of the group are highly satisfied with just one exception, a user that has low satisfaction scores, the group satisfaction score will still be high and is probable that the least satisfied user will be ignored. This observation leads us to define a new user utility score, namely the user disagreement. We define the disagreement of a user u as the difference between the u's satisfaction score and the maximum satisfaction score among the group members.

$$userDis(u, GL^j_G) = max_{\forall_{u_l \in G}} sat(u_l, GL^j_G) - sat(u, GL^j_G). \quad (8)$$

This allows us to better determine if a group member is systematically being favored (the user will have very low user disagreement scores) or is disregarded (the user has very high user disagreement scores).

We can generalize this in a group score by considering at each recommendation round the lowest and highest satisfaction scores among the group [12].

$$groupDis(GL^j_G) = max_{u \in G} sat(u, GL^j_G) - min_{u \in G} sat(u, GL^j_G). \quad (9)$$

Subsequently, we define the overall group disagreement of G for the entire recommendation sequence as:

$$groupDisO(\mathcal{GR}) = max_{u \in G} satO(u, \mathcal{GR}) - min_{u \in G} satO(u, \mathcal{GR}). \quad (10)$$

As a second alternative reward of the SQUIRREL model, we can utilize the harmonic mean of *groupSatO* and *groupDisO*, namely their F-score. Considering the input functions that F-score needs, we use $1 - groupDisO$ to simulate the group agreement.

$$R_{sd}(\mathcal{GR}^j) = 2\frac{groupSatO(\mathcal{GR}^j) * (1 - groupDisO(\mathcal{GR}^j))}{groupSatO(\mathcal{GR}^j) + (1 - groupDisO(\mathcal{GR}^j))}. \quad (11)$$

Overall, this strategy consists of two components, *groupSatO* and *groupDisO*, reflecting the degree to which an item is preferred by the members of the group and the level at which the members disagree or agree with each other, and targets an appropriate balance of these components.

In this work, we utilize these two different reward functions, namely the reward function R_s which is based on the overall satisfaction of the group and the reward function R_{sd} which is based on the overall satisfaction and disagreement. However, the reward function is very flexible and can be altered to complement the specific purpose that the framework is used for.

3.3. Actions as aggregation methods

The actions are the driving force behind our model. The agent observes the state of the environment and the history of the rewards it has already achieve and decides to apply an action. The action will generate changes to the state which will then enable us to calculate a reward. Since the actions are the only mechanism that can effect change, they are the natural selection to implement the group recommendation process. As we have already mentioned, we employ a single recommender system to generate recommendation list for each group member. For our model, we consider this process as a black box. The group recommendation process can then be distilled down to aggregating these distinct recommendation lists into one group recommendation list. We consider different approaches for this with each one being a different action.

We consider the following 6 different aggregation methods for the SQUIRREL model. Specifically:

Average. It is the classic Average Aggregation method, where the group predicted score for an item is the average across all predicted scores for that item across all group members. In this case, all the predicted scores of an item for the group members are considered to be of equal importance.

RP80 [10]. This method combines group relevance and group disagreement scores. The authors define group relevance, $avg(d, G)$, of an item d to be the average prediction score that was produced from the single recommendation system across all the members of group G. They define the group disagreement, $dis(d, G)$ for item d to be Average Pair-wise Disagreement between the predicted scores of the group members. $dis(d, G) = \frac{2}{|G|(|G|-1)} \sum(|p(u_i, d) - p(u_l, d)|)$, where $u_i, u_l \in G$, and $u_i \neq u_l$. The Average Pair-wise Disagreement reflects the degree of consensus in the relevance scores for the data item d among group members. The final score for d for the group G is: $\mathcal{RP}80(d, G) = (1 - w) * avg(d, G) + w * (1 - dis(d, G))$, where w is a tuning parameter for the group relevance and disagreement. Overall, in this case, all the group members' predicted scores about an item are considered of equal importance and additionally, the method takes into account the group's disagreement about that item.

Par [11]. This method generates a group recommendation list by incrementally adding to it items that have the best combination of Social Welfare (SW) and Fairness (F) scores. The Social Welfare of an item is the average satisfaction of all members for that item. [11] uses a similar satisfaction measure as our own work (Eq. (5)), if the group recommendation list GL^j_G only consisted of one item, d. Thus, $SW(d, G) = groupSat(d)$. As Fairness, $F(d, G)$, it employs the variance over the member's satisfaction scores. The final score of an item for the group is: $\mathcal{PAR}(d, G) = \lambda * SW(d, G) + (1 - \lambda) * F(d, G)$. Overall, this method takes into account the average satisfaction that an item produces for the group members and balances it with the variance of the group members' satisfaction for that item.

SDAA [8]. The Sequential Dynamic Adaptation Aggregation method, at round j, dynamically computes a weight w, which is defined as the difference in the satisfaction scores in the previous round of recommendations between the most satisfied and the least satisfied group member, $w^j_{SDAA} = max_{u \in G} sat(u, GL^{j-1}_G) - min_{u \in G} sat(u, GL^{j-1}_G)$. This weight is utilized to compute the final group prediction score for each proposed item d, as a weighted sum between the item's average score across all group members at round j, $avgG(d, G, j)$ and the item d predicted score for the least satisfied user in the group: $score(G, d, j) = (1 - w^j_{SDAA}) * avgG(G, d, j) + w^j_{SDAA} * leastG(d, G, j)$. The function $avgG(G, d, j)$ returns the average score across all group members for item d at round j. The function $leastG(G, d, j)$ returns the score for d for the least satisfied user at the beginning of round j. That is, the least satisfied user after round $j - 1$. When $j = 1$, i.e., the first recommendation round, the aggregation function turns into a simple average aggregation function. Overall, this method takes into account the past satisfaction of the group members to calculate a weight. This weight balances the average predicted score of an item for the group and the least satisfied member's predicted score for that item.

SIAA [8]. Similarly to SDAA, the Sequential Individual Adaptation Aggregation method also utilizes a weight for aggregations. In contrast to SDAA, which considers the group as an entity, SIAA focuses on each group member individually. For each member, it calculates a weight based on their overall satisfaction and the user disagreement of the previous iteration: $w^j_{SIAA}(u) = (1 - b) * (1 - satO(u, \mathcal{GR}^{j-1})) + b * userDis(u, GL^{j-1}_G)$, where b is the weight we use to balance the overall satisfaction and user disagreement scores. Given that we are in the jth round of recommendations, $\mathcal{GR}^{j-1}$ expresses the recommendations of all the previous $j - 1$ iterations. We apply this weight directly to the predicted score for each item after the single recommendation process is done. The final group prediction score for an item is the average over all group members' weighted scores. Overall, this method calculates a weight for each group member and applies it during the aggregation phase. The weight is based on the overall satisfaction of the user and their satisfaction on the previous round of recommendations.

Avg+ [8]. From evaluations performed in previous works, we observed that the classic average aggregation function offers one of the best group satisfaction scores, but simultaneously one of the worst group disagreement scores. In order to counter this drawback, we propose the Avg+ aggregation method that consists of two phases. In the first phase, we employ an average aggregation. Then, in the second phase, we iteratively populate the group recommendation list, with items that generate the minimum possible group disagreement score: $GL^j_G = GL^j_G \cup \{d \mid min_{\forall d \in AvgList^j_G}(groupDis(GL^j_G \cup d)) \vee d \notin GL^j_G\}$, where $AvgList^j_G$ is the list of the top k items after the first phase of the aggregation.

In this work, we study the above mentioned aggregation methods as our actions in the model, however they can be augmented by any additional method.

3.4. Problem formulation

Given the state, actions and reward we have defined, the purpose of our model is the following. At the jth round of recommendations, the model is given as input a group G and a set of recommendation lists B^j_u, for all $u \in G$ generated by a single user recommendation system for the current round. The model consists of the following: (1) a state S - each group member overall satisfaction up to the $j-1$ round, (2) a set of actions A - the different aggregation methods that the model has available, and (3) a reward function R - either based on the overall group satisfaction or the combination of group satisfaction and group disagreement.

At the jth round and all former and subsequent rounds of recommendations, the model considers the state of the group, and selects an action a that maximizes the expected chosen reward. Then, the selected action is applied to aggregate the set of recommendation lists B^j_u to a group recommendation list GL^j_G and further transition the state from s^j to s^{j+1}.

4. Experimental set up

4.1. Datasets

For the evaluation section of our work, we did not have access to datasets that contain interactions between groups and a system, where for example, the group as an entity has rated an item.[1] In lieu of such a dataset, we artificially created groups based on information taken from three real datasets. The 20M MovieLens Dataset [14], the 15M book reviews from GoodReads [15], and the Amazon dataset [16]. MovieLens contains 20M ratings across 27,3K movies given by 138,5K users between 01.1995 and 03.2015. The GoodReads dataset contains around 15M ratings from 465K users for about 2M books. Finally, the Amazon dataset contains 18M reviews given by 1.1M users spanning from May 1996 to July 2014.

Since we want to evaluate our model for a sequence of recommendations rounds, we need to simulate a time flow, where between the rounds some time has passed. This means that the recommender does not start with all the data available, but the information is sequentially augmented after each round. To simulate this, we first sort each dataset chronologically by the time that each rating was given. Then we split the datasets into chunks and after each round a new chunk is introduced to the system.

MovieLens Dataset. Initially, we divide the dataset almost evenly into two parts. The first part is the starting dataset of the system, and contains the chronologically first ratings, which were given between 01.1994 and 12.2003. This initial dataset consists of 8.381.255 ratings, 73.519 users and 6.382 movies. The reason we initiate the system with such a large dataset is to avoid the cold start problem.[2] The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of 6 months. Of those 22 chunks, we utilized the first 14 for our experiments.

GoodReads Dataset. Due to the difference in the distribution of ratings in the GoodReads dataset, we used a slightly different splitting process, since we have very few ratings in the first years and numerous in the latter. The initial dataset for the MovieLens was almost the 40% of the whole dataset. We initiate the system with a proportional sized data for the GoodReads, which translates into around 6.296.000 ratings. We split the rest of the dataset into chunks of equal size, which contain 674.565 additional ratings each.

Amazon Dataset. An additional pre-processing step is needed for the Amazon dataset. Unlike the MovieLens and GoodReads datasets, in the Amazon dataset the minimum number of ratings that a user has reported is five (5), whereas in the former are twenty (20). We further prune the dataset and exclude all the users that have rated less than 20 items. This results in a dataset that consists of 10.5M ratings from 180.118 users for 722.003 items. We split the Amazon dataset following the same process as the GoodReads dataset. The initial Amazon set consists of roughly the 40% of the overall dataset, which is translated to roughly 4M ratings. The rest of the dataset is split equally to 14 chunks. Finally, unlike MovieLens and GoodReads, Amazon contains duplicate ratings, meaning that the same user has rated the same item more than once. We only keep the most recent rating in this case.

All three datasets are split evenly in terms of their overall size. The initial chunk is equivalent to their 40% of the entire size and the rest 60% is split into 14 chunks. However, all three datasets differ in terms of their sparsity (i.e., how many ratings are missing over $|I \times U|$). Fig. 2 shows the sparsity of all three datasets with C1 being the initial chunk. The sparsity of each subsequent chunk is computed after it has been augmented into the system, with C15 being the entire dataset. The sparsity of the datasets increases after the inclusion of each chunk, because we insert into the system additional new users and items with relatively low number of ratings that correspond to either (i.e., the number of ratings given by a new user and the number of ratings a new item receives). It is worth noting that the GoodReads and Amazon datasets are far more sparse than the MovieLens, leading to have an adverse effect on the quality of recommendations (we discuss this in Section 5.4).

4.2. Group formation

Our goal is to evaluate our model using some real life scenarios. Since we do not have information, for example, which users are friends or have liked a review given by a user, we presume that if two people share the same interests, then they have similar ratings for the same data items. We want to simulate two different real-life scenarios. First, when a new person enters an established group, and their interests may very well be dissimilar to the rest of the group. For example, consider a new friend that joins an already established group in a movie nights series. Second, consider the case when random people join together for

[1] An experimental method that can be used for observing the group decision-making process appears in [13].

[2] The cold start problem in recommender systems appears when there is not enough information about a user and the system is unable to propose to him/her relevant items. This problem is beyond the scope of this research.

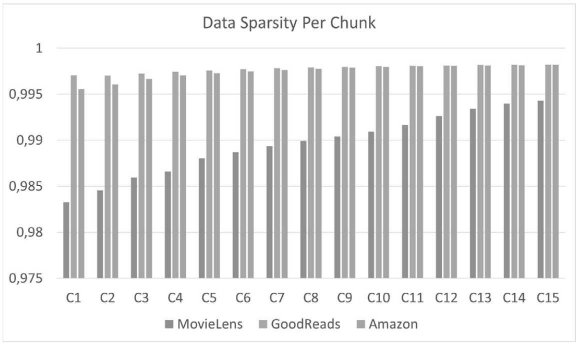


Fig. 2. The sparsity of the three datasets, MovieLens, GoodReads and Amazon, when each chunk is augmented into the system.

an activity, like a book club, where each member has their own tastes, however, they will have to read the same book.

We evaluate our model on these two types of groups based on the different degree of similarity shared between the group members. We utilize only the initial datasets from the splitting process to calculate the similarity scores and form groups. We use the Pearson Correlation [17] similarity function, which takes values from -1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity: $s(u_i, u_l) = \frac{\sum_{d\in X}(r(u_i,d)-\bar{r}_{u_i})(r(u_l,d)-\bar{r}_{u_l})}{\sqrt{\sum_{d\in X}(r(u_i,d)-\bar{r}_{u_i})^2}\sqrt{\sum_{d\in X}(r(u_l,d)-\bar{r}_{u_l})^2}}$, where X is a set that consists of the items that are rated by both users, $r(u, d)$ is the rating that user u gave to item d and $\bar{r}_u$ is the mean of u's ratings.

We regard two users to be similar if they have an 0.5 similarity score or above, and dissimilar when they have -0.5 or lower similarity score. The types of groups we are considering are the following:

4 similar – 1 dissimilar (4+1): The four members of the group are similar to each other, while the single member is dissimilar to all the rest.

5 dissimilar (5 Diss): All members of the group are dissimilar with each other.

4.3. Single user recommendations

In our evaluation, we derive the group recommendation list by combining the individual recommendation lists of each group member. We consider this first step, the single recommendation, as a black box, and our system recognize the individual recommendation lists of each user as that user's ground truth, since this is the only information that it has for that user. The actual single user recommendation system can be any system available, and is not restricted in any fashion beyond the fact that it needs to generate a recommendation list for a user. Systems that generate only one item can be used, but then the aggregation phase is rather trivial.

For this work, we use a user-based Collaborative Filtering (CF) recommender [18]. It generates a recommendation list by first finding similar users and then exploiting their likes and dislikes to propose items. The similarity function we used in CF is once again the Pearson Correlation. We consider two users similar if they share a similarity score greater than a threshold and they have given a rating to at least 5 common items. Due to the difference in each dataset's semantics, we set different similarity thresholds for each dataset. The MovieLens similarity threshold is 0.8, and for the GoodReads and Amazon dataset is 0.7.[3] Given a user u, to predict preference scores for item d, we use the Weighted Sum of Others Ratings [19], $p(u, d) = \bar{r}_u + \frac{\sum_{u'\in(P_u\cap U(d))} s(u,u')(r(u',d)-\bar{r}_{u'})}{\sum_{u'\in(P_u\cap U(d))} |s(u,u')|}$. Since a user may have many similar other users, we only consider the top 100 most similar to them denoted as P_u and $U(d)$ describes all the users that have rated item d.

5. Experimental results

5.1. Preliminaries

In our experiments, at each round, we propose to the group a list of 10 recommended items. Given that the group is aware of all data items presented to the group in a previous round, we do not recommend these items again.[4]

We also use the two format of groups that correspond to two life scenarios. (1) 4 similar – 1 dissimilar **(4+1)**, and (2) 5 dissimilar **(5Diss)**. For each type, we generate 100 different groups with five members from each dataset, MovieLens, GoodReads and Amazon. From those, we use 80 groups as a training set for our model and the remaining 20 groups as the test dataset. Additionally, we use the test dataset to individually evaluate all the aggregation functions we use as actions, namely Average, SDAA, SIAA, Avg+, RP80 and Par.

Training the model for only a specific group type is rather limiting since it will force the model to learn only that model. However, there is a case to be made about the need of such an approach. For example, if there is an application that groups people randomly for an activity for more varied social interactions, then a system that specializes in balancing the different needs of people that are dissimilar to each other is more advantageous. Nonetheless, we want our model to be able to be utilized in more universal conditions. To achieve this, we randomly selected 40 groups from the training sets for each group type and 10 from their corresponding test sets. This generated three additional training and test sets, one for each dataset, MovieLens,GoodReads and Amazon, with 80 groups in the training set and 20 groups in the test set, denoted as the **AllGroups** test set. To avoid overtuning the model for a specific group type, during the training phase we randomized the order of the groups.

[3] Due to the fact that MovieLens is more dense than GoodReads and Amazon, we selected a higher similarity threshold for it, enabling Pearson Correlation to return approximately the same number of similar users for all datasets.

[4] Alternatively, we can imagine that the group is presented with one item only. Given that we target at the group consensus, the prediction model regarding the group choice for this item is a worth studying problem without a straightforward solution that we leave it for future work.

To create our SQUIRREL RL model, we use the Tensorforce library,[5] with the Proximal Policy Optimization (PPO) as our learning policy, with probability to transition from state s to state s' under a random action is set to 0.3, while the γ parameter is set to 0.99. Since each dataset is fundamentally different from the other, notably the GoodReads and Amazon datasets are far more sparse than the MovieLens (Fig. 2), we experimentally found the best learning rate, namely, the speed at which the model *learns*, for each dataset individually. This was done through extensive experiments that we do not present due to space constrains. We used 0.0022 learning rate for the MovieLens dataset and 0.0020 for the GoodReads and Amazon.

Furthermore, each individual aggregation method has further parameters to tune. For the RP80 aggregation function, we choose to set the variable w to 0.8, since it performs better based on the experiments presented in [10] for the MovieLens dataset, which we also use. This weight is used in balancing the group relevance and group disagreement. According to [11], the Par method uses $\lambda = 0.8$ to combine the social welfare and fairness scores. In [8], for the SIAA aggregation function, we empirically found, using the same datasets, the best value for variable b, which is used to combine the overall satisfaction and disagreement of a user, is 0.2. Finally, in the same work, for Avg+, we found that the best results are given when $k = 50$, where k is the number of items with the highest prediction score.

5.2. Measures

At each round, we propose to the group a list of 10 recommended items. At the end of each recommendation round, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDisO(G, \mathcal{GR})$ for all the groups in the test set. These measures are used to calculate the reward functions R_s and R_{sd}.

To further analyze the methods' quality, we use as secondary measure the Normalized Discounted Cumulative Gain (NDCG), where we want the best possible items for a user to appear with the highest ranking possible in the group recommendations: $NDCG(u, G, j) = \frac{DCG(u,G,j)}{IDCG(u,G)}$, with $DCG(u, G, j) = \sum_{d \in GL_G^j} \frac{|d \cap B_u^j|}{log_2(r(d, GL_G^j)+1)}$, where $r(d, GL_G^j)$ is the rank of item d in the group recommendation list GL_G^j. IDCG is the best possible outcome for user u_i. We supplement the results from NDCG, with the Discounted First Hit (DFH) metric that counts if a user has seen an item that is highly relevant to him/her in the early ranks of the group recommendation list: $DFH(u, GL_G^j) = \frac{1}{log_2(fh+1)}$, where fh is the rank of the first item that appears both in the group recommendation list and the individual preference list of the user. Since NDCG and DFH refer to one user, we apply them to the group by computing them for each member and then taking the average over them, as recent papers on group recommendations do (e.g., [20]). We do this at the end of each round.

5.3. Analyzing the actions selected

In Tables 2 and 3, we present the actions that the model selected during both the training and testing phase for both versions of the reward function. Table 2 presents the actions selected when utilizing the reward function R_{sd}, which is based on satisfaction and disagreement (Eq. (11)), while Table 3 shows the actions selected when we use the satisfaction reward function R_s (Eq. (7)).

With either reward function SQUIRREL selects the most appropriate aggregation to apply, based on what maximizes that reward function. In more detail, in Table 2 showcasing the actions tested and selected for the reward function R_{sd} for the MovieLens dataset, we can observe that the model has predominantly chosen the SDAA aggregation function with minimal exceptions. We can explain this choice with Figs. 4–6, where we show the R_{sd} values for all aggregation methods for the MovieLens dataset. SDAA is the best action for the model to take, since it produces the best R_{sd} scores due to its lower group disagreement scores in the latter rounds.

Table 2
Actions chosen during the training and testing phase for all group formats, while utilizing the reward function based on satisfaction and disagreement, R_{sd}.

MovieLens Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	50	0	70	0	69	0
SDAA	869	280	917	280	387	280
SIAA	69	0	47	0	317	0
Avg+	122	20	105	20	315	20
Par	71	0	54	0	93	0
RP80	19	0	7	0	19	0
GoodReads Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	28	0	7	0	165	0
SDAA	22	0	35	0	24	0
SIAA	65	20	227	20	216	20
Avg+	860	280	676	280	590	280
Par	190	0	51	0	144	0
RP80	35	0	204	0	61	0
Amazon Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	149	0	193	0	127	0
SDAA	64	0	222	0	106	0
SIAA	333	240	356	240	504	240
Avg+	242	60	296	60	216	60
Par	199	0	56	0	87	0
RP80	213	0	77	0	160	0

It is also worth to examine the number of times the model did not select SDAA. For all test sets, the model does not select the SDAA function in favor of the Avg+ function, exactly 20 times (see Table 2). These numbers correspond with the number of groups present in each test set. This is significant because the R_{sd} scores for the first round are comparable between many aggregation functions but the Avg+ function perform slightly better since it has lower group disagreement scores. Again we can corroborate this, when we examine the R_{sd} scores generated by all aggregation methods shown in Figs. 4–6. This is because two of the proposed aggregation functions, SDAA and SIAA, do not have the necessary information during the first round of recommendations. SDAA defaults to a classic Average for the first round and SIAA utilizes the user disagreement of the previous round, something that does not exist during the first round. So both of them default to a classic average aggregation for the first round. On the other hand, Avg+ does not have any such drawback. The model is able to identify this, and selects the better performing aggregation function for the first round.

SQUIRREL behaves similarly when utilizing the GoodReads dataset as well. As we can observe in Figs. 8–10, the Avg+ aggregation function offers the best R_{sd} scores out of all the aggregation methods used as actions. However, during the first round, many aggregation functions share the same R_{sd} scores, so the system has selected a different action for that round. Amazon presents similar results, where now the aggregation that produces the best

[5] https://tensorforce.readthedocs.io/en/latest/.

R_{sd} scores is SIAA with the exception of the first three rounds where Avg+ is better, as it is shown in Figs. 12–14. This is why we can see in Table 2 that the Avg+ aggregation method is selected 60 times during the test phase.

Finally, Table 2 also shows that the actions that were selected during the testing phase for the reward function R_{sd} remain the same across all different training sets (i.e., 4+1 and 5Dis). Subsequently, the same action is selected when we train the model using the **AllGroups** training set that contains all different formats of groups.

Table 3 shows the actions taken during the training phase and selected by the SQUIRREL model during the test phase when utilizing the reward function R_s. As we can see in Fig. 3[a,b,c]-left for MovieLens, Fig. 7[a,b,c]-left for GoodReads, and Fig. 11[a,b,c]-left for Amazon, the satisfaction generated by the aggregation functions are approximately the same with two notable exceptions. First, RP80 is designed to offer the best possible group disagreement in lieu of the group satisfaction, thus it has lower satisfaction scores. Second, Avg+ is based on the Average method, which offers one of the best group satisfaction scores. However, Avg+ also considers sub-optimal items (in regards to the group satisfaction score they produce) so as to discover items with also good group disagreement.

The similar satisfaction scores of the various aggregation methods make it difficult for our model to select an aggregation function that is universally good for each dataset. This is because R_s considers just the overall group satisfaction, and multiple aggregation functions (i.e., actions) produce very high satisfaction scores. This results in different functions to be selected for the various training sets. In contrast, R_{sd} is more refined, as it considers both group satisfaction and group disagreement, and is able to identify the most effective aggregation function for each dataset (see Table 2).

Table 3
Actions chosen during the training and testing phase for all group formats, while utilizing the satisfaction reward, R_s.

MovieLens Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	83	20	48	0	130	0
SDAA	22	0	90	20	132	0
SIAA	366	0	26	0	140	280
Avg+	7	0	25	0	553	0
Par	701	280	940	280	187	20
RP80	21	0	71	0	58	0
GoodReads Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	32	0	20	0	319	269
SDAA	229	20	278	20	159	11
SIAA	135	0	62	0	143	0
Avg+	35	0	40	0	233	20
Par	659	280	772	280	158	0
RP80	110	0	28	0	188	0
Amazon Dataset						
	4+1		5Dis		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	412	220	237	20	533	270
SDAA	47	0	166	0	96	0
SIAA	163	0	517	280	184	0
Avg+	59	0	27	0	129	0
Par	382	80	192	0	235	30
RP80	137	0	61	0	23	0

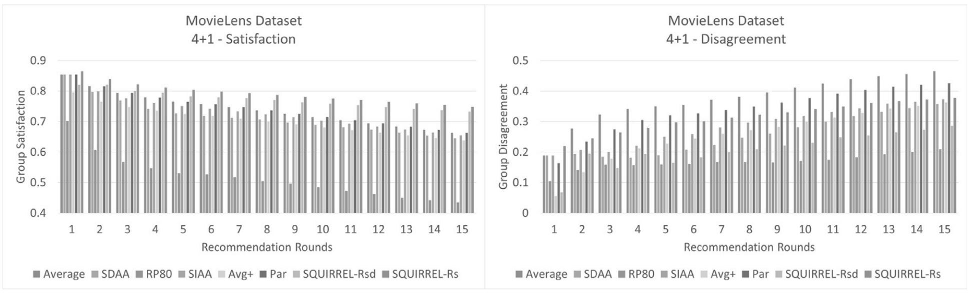
5.4. Analyzing the satisfaction and disagreement scores

The SQUIRREL model primarily selects one aggregation function, the one that maximizes the selected reward function, but is also more flexible during the first round of recommendations. Due to this small change that the model makes, the primary aggregation function (the one that the model uses for the rest of the rounds) performs differently. For example, when we compare the results in Fig. 3a that shows the group satisfaction and group disagreement for the 4+1 test set for the MovieLens dataset, and Fig. 4 that shows the R_{sd} scores for all aggregation methods, we can see that SQUIRREL correctly identifies that the primary aggregation function (in this case the SDAA) under-performs in the first round, so it selects the best one which is the Avg+. In this case, SDAA underperforms in the first round of recommendations, since we do not have the previous disagreement of the users and so the aggregations defaults to the classic Average. This selection of a different aggregation method in the first round has a cascade effect on the results for the rest of the rounds. We can observe a similar behavior for the rest of the test sets.

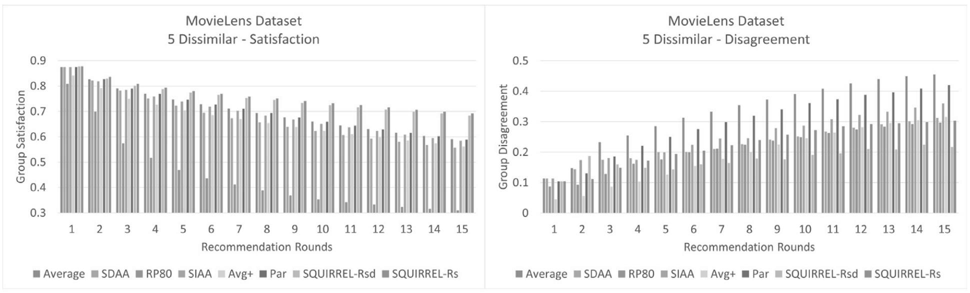
In general, the small change on the first round, makes the model perform better for the rest of the rounds both for group satisfaction and group disagreement. In terms of group satisfaction SQUIRREL offers the best group satisfaction scores. This difference is mainly due to first round of recommendations, where as already stated, the model chooses a different aggregation function than the one typically used in the rest of the recommendation rounds. Thus, it now has a different pool of items to recommend to the group in the next rounds. Subsequently, this makes it behave differently than the aggregation method that is primarily used in the rest of recommendation sequence. This is more apparent in the latter rounds of recommendations, where SQUIRREL is able to perform better than the rest of the aggregation methods.

When comparing the results between the two reward functions, for example the last two bars in Fig. 3[a] for the MovieLens dataset and the 4+1 training set, the R_s reward function offers better group satisfaction scores (Fig. 3[a]-left), since that reward function maximizes group satisfaction. However, it generates worse group disagreement compared to R_{sd} (Fig. 3[a]-right). This is expected, since the R_{sd} considers both group satisfaction and disagreement. When we compare the rest of the aggregation functions, for group satisfaction SIAA, SDAA, Par and Average have very comparable results, while the Avg+ falls slightly behind since it also considers sub-optimal items in terms of the group satisfaction they generate. On the other hand, RP80 offers very low group satisfaction scores, but one of the best group disagreement scores. This is because, it was designed to minimize the disagreement score. Additionally, RP80 has better group disagreement in more homogeneous group formats like 4+1 while it falls slightly behind in 5Diss test set. SDAA and SIAA have very good group satisfaction but average group disagreement scores. On the other hand, Average and Par have very good group satisfaction scores but also very high group disagreement.

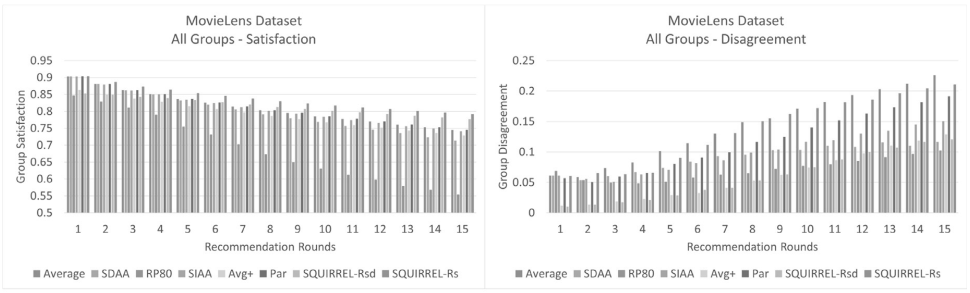
We can observe that the aggregation methods have different behaviors in the three different datasets. For example, the best performing one for MovieLens is SDAA, for GoodReads is Avg+ and for Amazon dataset is SIAA. The discrepancy of performance between the three datasets is because the difference in their sparsity, namely GoodReads and Amazon datasets are more sparse than the MovieLens. This negatively affects the single recommender system and reduces the number of items that are highly relevant to each individual group member. On the one hand, in the MovieLens dataset which is the most dense dataset of the three (see Fig. 2), the single recommender system generates more relevant items per group member. This benefits the SDAA aggregation, since it directly balances the group satisfaction and group disagreement. Thus the more common items that are relevant to most group members the more beneficial it is for the aggregation method.



(a)



(b)



(c)

Fig. 3. MovieLens Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

On the other hand, the sparsity of the GoodReads and Amazon datasets benefits the Avg+ and SIAA aggregation methods. A lower number of highly relevant items per user means that during Avg+'s first phase where it selects the top 50 items with the highest Average score, all the best possible items for the group are considered. Then, Avg+ can select the items that produce the best group disagreement. SIAA only takes into consideration each individual group member's satisfaction and disagreement scores and not the group's, thus is better equipped to handle the sparsity of the datasets. The SQUIRREL model is able to adjust to the different datasets and outperform all the individual aggregation methods.

Finally, we can see from Figs. 3, 7 and 11 that for all test scenarios the performance of all aggregation methods is reduced after each round of recommendations. This is because of two main reasons. First, as Fig. 2 shows, the sparsity of all datasets is increased after each additional chunk of data is augmented in the system. Second, after each round, we eliminate the top 10 most suggestions for the group, i.e., these items cannot be recommended again.

5.5. Quality of recommendations

In Table 4, we show the NDCG and in Table 5 the DFH scores for all aggregation functions, as well as, for the SQUIRREL model when utilizing the two reward functions. We present the average scores after the end of all 15 rounds of recommendations. The NDCG scores in Table 4 reflect how good the recommendations were for the group, i.e., how many items in the group recommendation list are also included in the user's individual

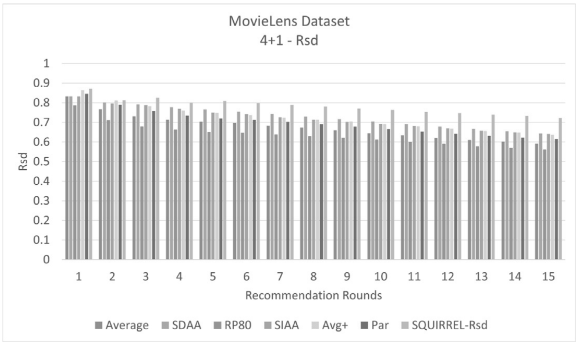


Fig. 4. MovieLens Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

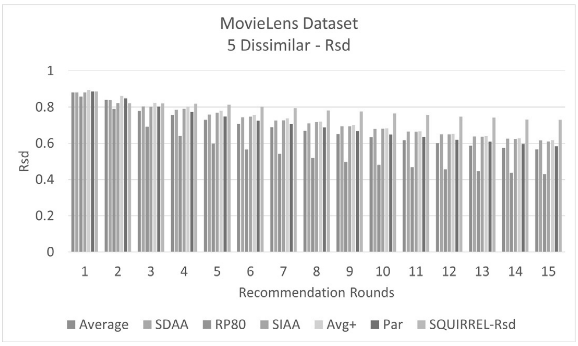


Fig. 5. MovieLens Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

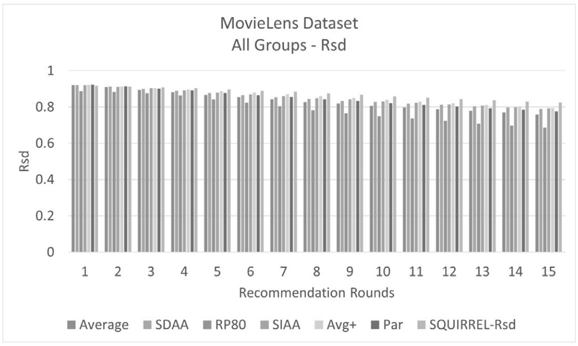
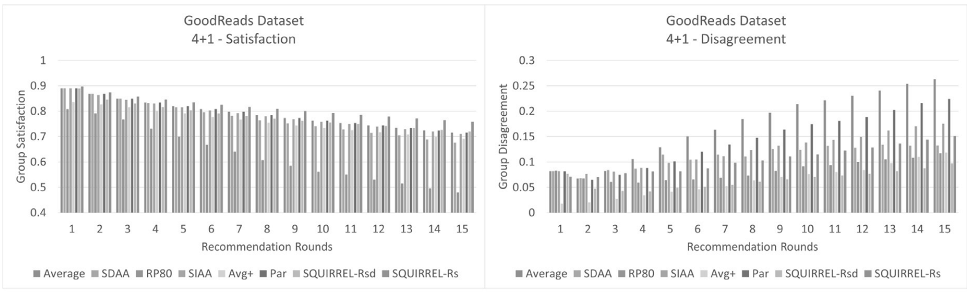


Fig. 6. MovieLens Dataset: R_{sd} scores for all aggregation methods under the AllGroups test scenario.

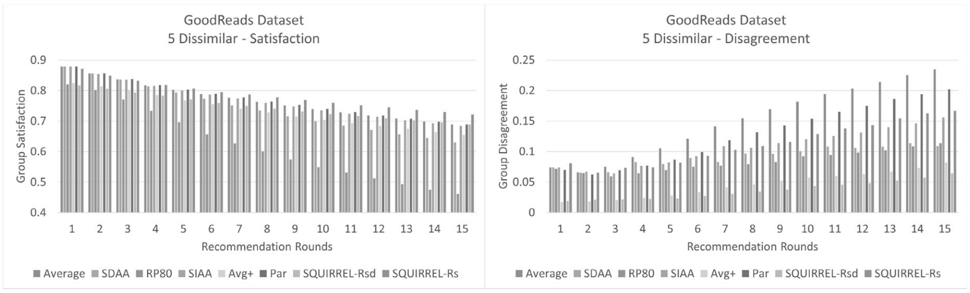
recommendations. The DFH scores in Table 5 describe how relevant at least one item in the group recommendation list is for the users.

In Table 4, we see that the SQUIRREL model's higher NDCG values mean that it is able to identify more relevant items for the groups. Also it recommends items that are highly ranked by each user, given that the DFH scores are high. In contrast, the Avg+ aggregation method has lower NDCG scores, because it also considers items that may not be highly relevant to each user. This is further corroborated by the group satisfaction scores shown in Figs. 3-left, 7-left and 11-left. The aggregation methods that have high satisfaction scores, like Average, SDAA, SIAA, Par and both SQUIRREL models also have high NDCG scores. This is because both the group satisfaction score and NDCG describe the same general principle of how relevant are the items in the group recommendation list for each user.

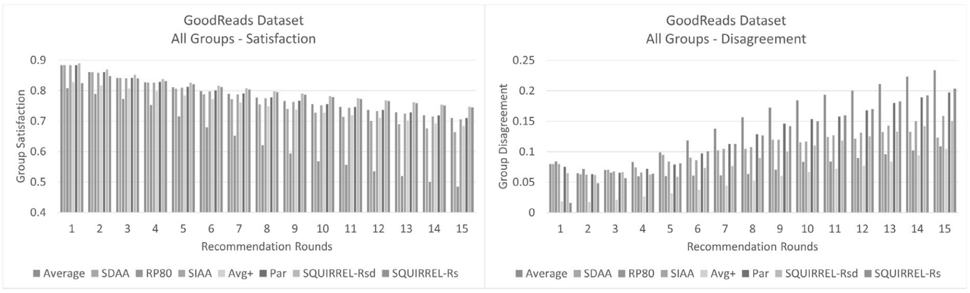
Finally, to better visualize the results, Figs. 15 and 16 show the NDCG and DFH values, respectively, per round. We showcase the results for the MovieLens dataset and the 5Diss test scenario. These values are the average for all groups in the 5Diss test set per round. The reward function R_s has the best overall NDCG values since it also has the best group satisfaction (Fig. 3b-left).



(a)



(b)



(c)

Fig. 7. GoodReads Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

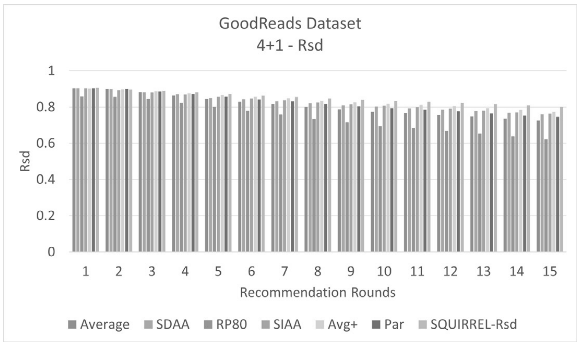


Fig. 8. GoodReads Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

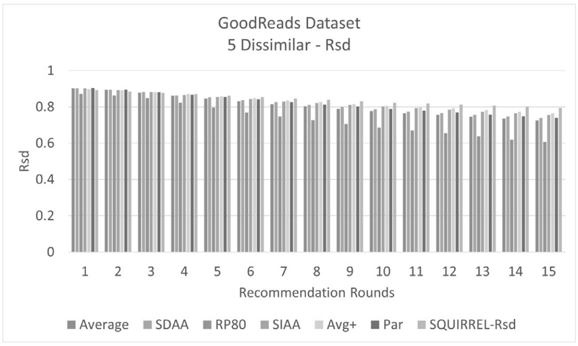


Fig. 9. GoodReads Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

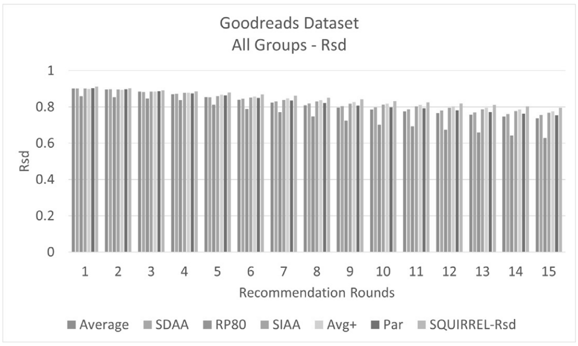


Fig. 10. GoodReads Dataset: R_{sd} scores for all aggregation methods under the AllGroups test scenario.

Table 4
NDCG values for all datasets, MovieLens, GoodReads and Amazon and all test scenarios, 4+1, 5Diss and AllGroups.

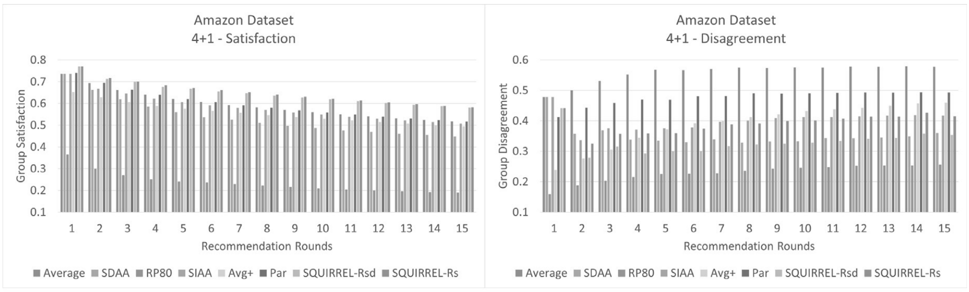
MovieLens Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.038	0.041	0.038	0.033	0.039	0.021	0.054	0.052
5Diss	0.043	0.044	0.043	0.036	0.043	0.017	0.054	0.060
AllGroups	0.042	0.044	0.041	0.034	0.042	0.019	0.058	0.057
GoodReads Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.041	0.053	0.043	0.032	0.042	0.020	0.046	0.063
5Diss	0.042	0.057	0.042	0.031	0.043	0.017	0.040	0.065
AllGroups	0.042	0.053	0.042	0.030	0.042	0.020	0.043	0.066
Amazon Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.028	0.047	0.035	0.025	0.033	0.003	0.044	0.026
5Diss	0.030	0.052	0.031	0.024	0.030	0.005	0.029	0.024
AllGroups	0.032	0.056	0.036	0.030	0.035	0.003	0.052	0.043

Additionally, it is more apparent that during the first round of recommendations all aggregation methods have a higher performance which is then significantly reduced in the following rounds. As already stated, this is because, as shown in Fig. 2, the sparsity of all datasets is increased after each chunk is augmented in the system, as well as, because after each round we eliminate from consideration the top 10 items for each group.

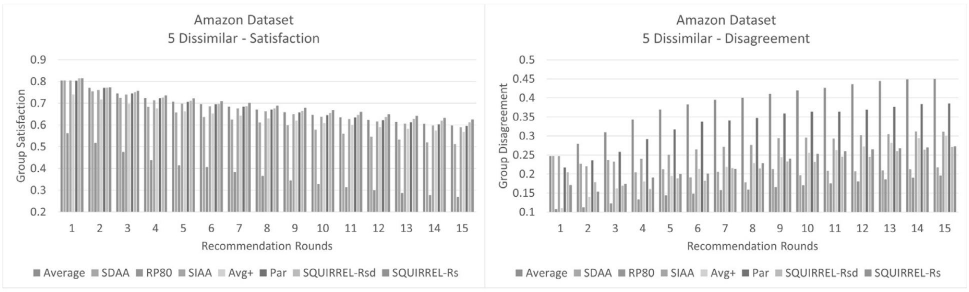
5.6. Group size evaluation

For all the previous evaluations, the group size was set to five. Next, we evaluate our model when the group size is increased. We selected the MovieLens dataset to experiment with since its higher density allowed us to form larger groups. We expand on the 5 dissimilar test scenario and introduce two new test cases, 7 dissimilar (**7Diss**) and 9 dissimilar (**9Diss**). Fig. 17a shows the group satisfaction (left) and group disagreement (right) for the 7Diss test scenario. Fig. 17b shows the respective scores for the test scenario 9Diss.

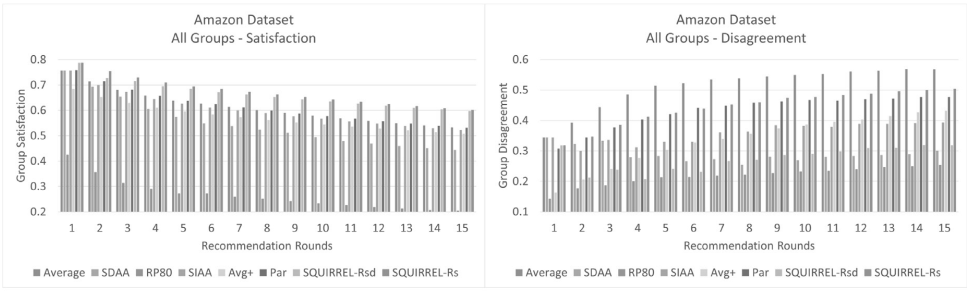
The performance of all aggregation methods lowers when the group size is increased. This is expected since it is more difficult to satisfy a larger group. However, SQUIRREL, overall, is able to more effectively handle a large number of dissimilar group members. It is largely unaffected by the increased group



(a)



(b)



(c)

Fig. 11. Amazon Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

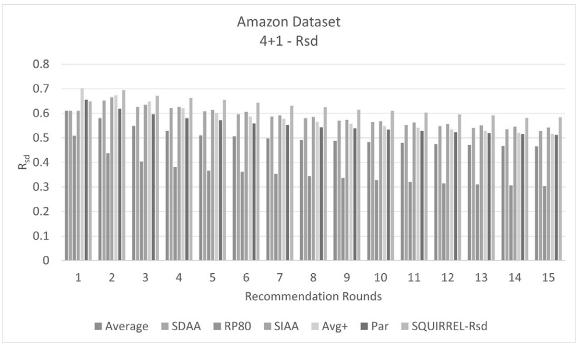


Fig. 12. Amazon Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

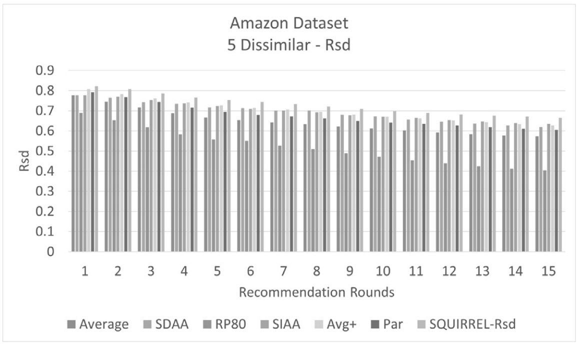


Fig. 13. Amazon Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

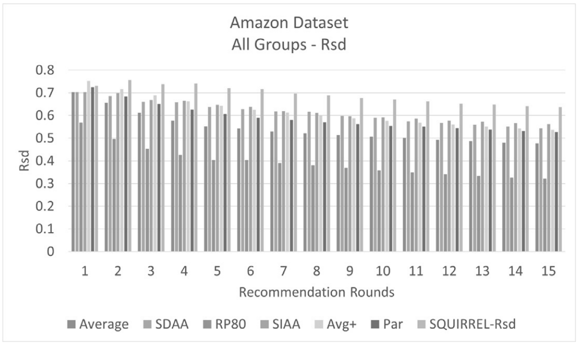


Fig. 14. Amazon Dataset: R_{sd} scores for all aggregation methods under the AllGroups test scenario.

Table 5
DFH values for all datasets, MovieLens, GoodReads and Amazon and all test scenarios, 4+1, 5Diss and AllGroups.

MovieLens Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.900	0.868	0.897	0.936	0.913	0.810	0.923	0.937
5Diss	0.886	0.819	0.871	0.932	0.899	0.743	0.914	0.940
AllGroups	0.915	0.875	0.905	0.945	0.925	0.807	0.930	0.936
GoodReads Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.968	0.923	0.967	0.982	0.977	0.936	0.957	0.987
5Diss	0.962	0.893	0.958	0.979	0.968	0.929	0.954	0.973
AllGroups	0.976	0.929	0.975	0.985	0.982	0.955	0.964	0.985
Amazon Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.795	0.679	0.779	0.838	0.825	0.554	0.832	0.935
5Diss	0.886	0.755	0.876	0.916	0.903	0.714	0.890	0.935
AllGroups	0.820	0.677	0.804	0.860	0.844	0.579	0.837	0.872

size with the fall in performance between the 7Diss and 9Diss test cases being minimal. These results are further corroborated when we examine Figs. 18 and 19 showcasing the R_{sd} scores for the various aggregation methods. All aggregation methods have similar results in the first round of recommendation with SQUIRREL outperforming the rest in the latter rounds.

6. Related work

6.1. Group recommendations

Group recommendations have a significant research background [21]. In general, there are two main approaches for group

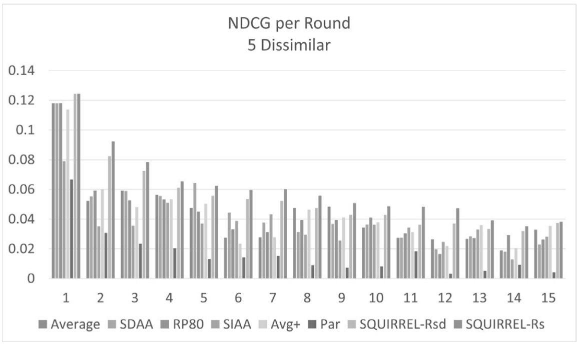


Fig. 15. MovieLens Dataset: NDCG values per recommendation round for the 5Diss test scenario.

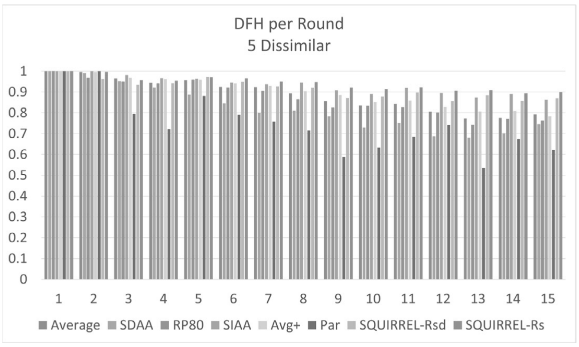


Fig. 16. MovieLens Dataset: DFH values per recommendation round for the 5Diss test scenario.

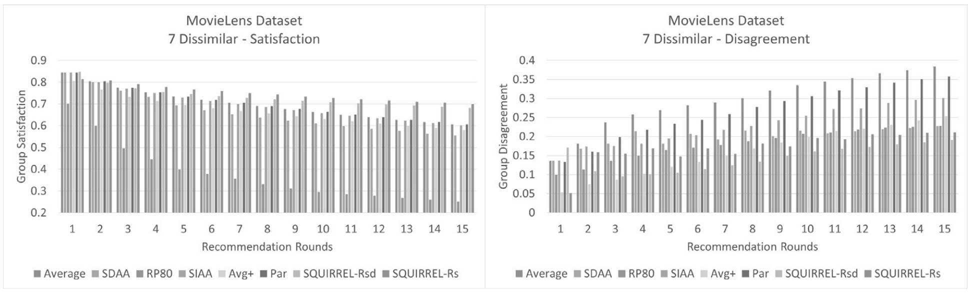
recommendations [22]. In the first approach (e.g., [23,24]), we combine the group members ratings to form a virtual user so that a standard recommendation approach can be applied. The second approach, and the most popular for group recommendations (e.g., [10,25,26]), is to employ a standard single-user recommender system and apply it to each individual group member. Then, we aggregate the group members lists into one single group recommendation list. In this work, we follow the second approach, since it is more flexible [27] and offers opportunities for improvements in terms of effectiveness.

In the aggregation stage, a group recommender system can take into consideration many different criteria. For example, work done in [28] suggests a group recommendation model which takes into account each individual group member's influence during the aggregation phase. They state that the more knowledgeable a member is about the items considered for recommendation, the more influence they have, meaning they have a higher weight during the aggregation phase. Additionally, [29] draws from recent advancements in attention network and neural collaborative filtering to deduce the aggregation strategy from the available data. In the same vein, [30] in addition to an attention mechanism, it also employs a Bipartite Graph Embedding Model (BGEM) to infer each member's influence to the group's final choice. [31] uses a preference-oriented social network and the social interactions among the group members, to finalize the group's choice without having access to complete preferences of the members.

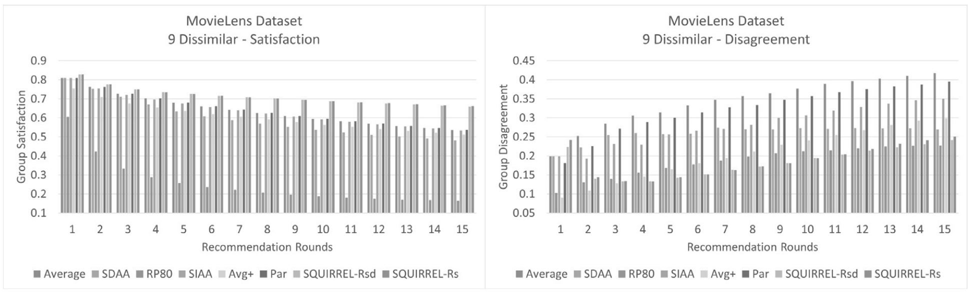
By observing how group members interact with one another, [32] determines the best aggregation strategy. These interactions were modeled as multiple voting processes in order to simulate how a consensus is reached, and a stacked social self-attention network was proposed to learn the voting scheme of the group members. In dividing a large group of people into subgroups based on their own interests, [33] offers a novel method of producing recommendations for a large group. Specifically, it identifies a set of potential candidate media-user pairs for each subgroup and aggregate the CF recommendations lists for each such pair. [34] proposes a two-phase group recommender that, tries to satisfy all the group members. In the first phase, they try to satisfy the whole group. In the second phase, they try to satisfy the members individually by filtering out irrelevant items to each member.

In [11], each group member is assigned a utility score based on how relevant the recommended items are to them. Then it balances the utility of the group members and generates a group recommendation list. In [20], the utility of a user is defined by the similarity between the individual and group recommendations of the user. Their approach involves considering sets of N-level Pareto optimal items when creating the group recommendation list. As part of the aggregation phase, [35] proposes a notion of rank-sensitive balance. As far as possible, the first recommendation should balance the interests of all group members. Similarly, the first two items together must also do the same; and so on.

Any of these rank aggregation methods can be modified to work on the SQUIRREL model as an additional action. Depending on the requirements of the method, supplementary changes should be made to the model, such as a different state or reward definition or additional input should be given to the model. Additionally, depending on the complexity of the aggregation method, the overhead of retraining them can be large; a complex group



(a)



(b)

Fig. 17. MovieLens Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods under the 7Diss and 9Diss test scenarios.

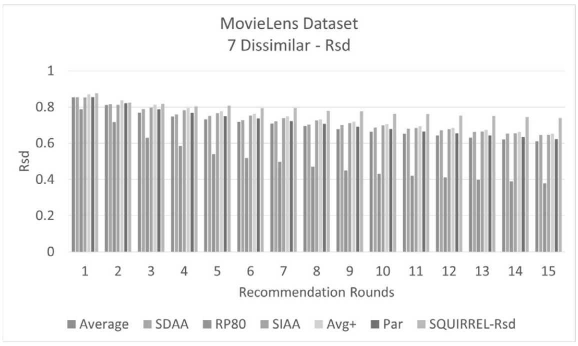


Fig. 18. MovieLens Dataset: R_{sd} scores for all aggregation methods under the 7Diss test scenario.

recommendation method needs more time for training than the simple Average aggregation method. In future work, we aim to further develop our model so as other than rank aggregation methods for group recommendation can be readily included as actions.

6.2. Sequential recommendations

It is generally considered that sequential recommenders fall into three broad categories, according to how many past interactions they take into account: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* [36]. The first approach considers only the most recent N user actions [37–39]. This is due to the amount of historical data logged by the system for the user – many of which are duplicates as well as not providing any useful information – and as a result the system is overwhelmed. The only interactions that are taken into account when making session-based recommendations are the ones that the user performed during the current session. The most common places to find them are in the news and advertisements [40,41]. In the last category, the system has information about the last interaction between it and the user, as well as the user's history. E-commerce and app recommendations often employ these recommenders [42–44]. Another session-aware music recommendation system is proposed in [2]. Based on a neural network architecture, users' preferences are represented as a sequence of embeddings, one for each session. The user's recent selections and session-level context (such as device usage and time) can predict the tracks a user will listen to. [3] proposes a multi-round recommender system using Variational

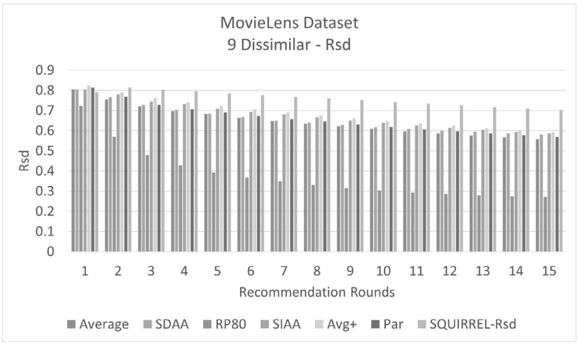


Fig. 19. MovieLens Dataset: R_{sd} scores for all aggregation methods under the 9Diss test scenario.

Autoencoders (VAEs) and introduces randomness into the regular operation of VAEs to achieve fairness [45] during multiple rounds, while in order to minimize bias and promote diversity, [4,46,47] penalizes scores given to items based on historical popularity.

The framework in [1] uses cross-neighbor relation modeling to uncover collaborative information using a bipartite graph, where the users and items are depicted as nodes and the interactions between them as the links. They not only consider the directly linked nodes but also the 2-hop neighbors as well, which they call high-order collaborative relations. They utilize them and both user-side and item-side historical sequences to capture user and item dynamics more effectively. The work in [48] proposes the GLS-GRL system, where an item–item co-occurrence graph captures user–item interactions in the entire history, as well as an item–item co-occurrence graph containing the same information for the current time period. As a result of graph representation learning, the GLS-GRL system is able to achieve long-term and short-term representations of users and subsequently merge them to obtain integrated representations of users. A constraint-based user-interactive attention mechanism encodes relationships between group members into group representations used for recommendations.

The work presented here, employs methods for sequential group recommendations, that were first proposed in [8,49]. SDAA considers the group as a whole, and dynamically calculates a weight based on the satisfaction of the group members. This weight is then used to combine two scores; the average preference score of an item for all group members and the preference score of the item for the user that is the least satisfied in the previous round of recommendations. In contrast, SIAA considers each group member individually. At each round it calculates a weight for each user based on the user's overall satisfaction and the user's disagreement in the previous round. Finally, Avg+ capitalizes on the advantages of the classic Average method while simultaneously tries to minimize its drawbacks. Overall, the previous works mentioned concern recommendation systems for single users.

Under specific application examples, [50,51] perform empirical research to investigate different aggregation strategies for suggesting a sequence of television items and music tracks, respectively, to groups of users. The proposed framework in this work is the first to focus on selecting an aggregation strategy, among a pool of available ones, for each round of group recommendations, relying on reinforcement learning.

6.3. Reinforcement learning in recommendations

In recent years, more and more research is focused in utilizing reinforcement learning for recommendations [52]. One of the first works in this domain was done in [53], where they propose a web recommender system, where the state of the environment is the last N pages that the user visited, actions are the page recommendations and reward is a weighted sum between the ranking of the recommended page and the time the user spend on that page. [54] proposes a DQN-based reinforcement learning framework for online personalized news recommendation. The framework is composed of two parts, offline and online. During the offline stage, the model is trained and during the online stage the agent produces a recommendation and logs the user feedback. After a period of time, the model enters again in the offline stage and depending on the logged feedback may make changes to the model. The system can accurately model the dynamic features of the news, in addition to the user preferences, in order to increase reward in the long term. In addition to click/no-click feedback, they also consider user return pattern in order to determine the user behavior, and employ an exploration strategy for more recommendation diversity. [55] employs reinforcement learning techniques to optimize the recommendation model for long-term accuracy of recommendations. They consider two main areas; cold-start and warm start. The model relied on the interactions between the recommender system (environment) and users (agents). This allowed it to be applied to environments with inadequate content information.

The work done in [56] proposes a list-wise recommendation framework based on deep reinforcement learning, a method that reduces redundant computation in scenarios with a large and dynamic item space. They train and evaluate the model offline, while applying the recommender system online. To successfully evaluate the model in offline stage, they simulate the response of a user (reward) using a user–agent interaction environment simulator, where the reward is calculated based on the similarity between the current state and the action taken with other historical data. [57] proposes a deep learning movie recommender model based on reinforcement learning, which utilize prioritized experience replay to depict the changes in the interests of the user over time. The system models the recommendation process as reinforcement learning, and in order to recommend movies based on user preferences, they use agents to learn about users' interests and movie features.

The work done in [58] proposes a user-side system for sequential music recommendation. It combines in a Markov Decision Problem (MDP) the users' explicit and implicit feedback. The explicit feedback consists of the users' music channel preferences and the implicit is generated by the users when they request a new music track. An MDP is also used in [59] where they propose a commercial system that utilizes an ordered sequence of selections for each user as the state of the environment, to predict and recommend a new item. However, since the e-commerce environment of this model is different than a classic recommendation system, several assumptions are made in order to be readily deployed as an application.

Each of these works offers a solution to the recommendation problem through reinforcement learning, but in most cases are centered around a specific domain of recommendations. In our work, we propose a framework that aspires to be more flexible in terms of the domain it can be applied and is able to combine different techniques to counter-balance the drawbacks that are inherently present in each recommendation solution.

7. Conclusion

In this paper, we propose the SQUIRREL framework, which enables sequential group recommendations via reinforcement learning. We treat the single user recommendation system as a black-box, and focus on aggregating the individual group members' recommendation lists into a group list. Specifically, we focus on producing group recommendations via rank aggregation, thus the actions that the model can take are various rank aggregation methods. The state of the model is the overall satisfaction of the individual group members, and we examine two different reward functions. The model is able to be further configured; additional rank aggregation methods can be applied, a different state can be defined and further reward functions can be considered. The model is able to dynamically select the most appropriate group recommendation method to apply depending on the state of the group.

We used three real-world datasets, 20M MovieLens, GoodReads and Amazon, to evaluate not only our proposed model but additionally, all the individual aggregation methods used as actions. The different semantics of each dataset alter the performance of all methods, but typically do not alter their performance comparative to each other. We train our model using different formation of groups, which depict different scenarios for group recommendations. Additionally, we combined the different group types into one test set to simulate a more general scenario. Although the actions selected for each group type are different, the overall performance of the SQUIRREL model did not change between the test cases. The SQUIRREL model is able to correctly identify the aggregation method that best maximizes the reward function utilized. We corroborate this with extensive evaluation of all methods used as actions in our model, and test it with two reward functions. Additionally, the model recognizes the drawbacks of certain aggregation methods and counters them by selecting different ones at the opportune moments. This has as a result to completely alter and enhance the performance of the primary aggregation method used. Finally, we evaluated the quality of the recommendations provided by calculating the NDCG and DFH values for the SQUIRREL model and all the aggregation methods used as actions. We show that SQUIRREL is able to provide quality recommendations, with one of the best NDCG and DFH scores.

In our future work, we want to extend our model by combing individual user and group satisfaction scores to define new states and rewards. Moreover, we want to expand the SQUIRREL model so as to be able to consider as actions other group recommendation algorithms in addition to the rank aggregation methods. This will require a restructure on the input that the model receives and further optimizations, depending on the complexities of the selected group recommendation methods.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] J. Qin, K. Ren, Y. Fang, W. Zhang, Y. Yu, Sequential recommendation with dual side neighbor-based collaborative relation modeling, in: WSDM, 2020.
[2] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, M. Lalmas, Contextual and sequential user embeddings for large-scale music recommendation, in: RecSys, 2020.
[3] R. Borges, K. Stefanidis, Enhancing long term fairness in recommendations with variational autoencoders, in: MEDES, 2019.
[4] R. Borges, K. Stefanidis, On mitigating popularity bias in recommendations via variational autoencoders, in: SAC, 2021.
[5] J. Masthoff, Group Recommender Systems: Combining Individual Models, Springer US, Boston, MA, 2011, pp. 677–702.
[6] K.J. Arrow, A difficulty in the concept of social welfare, J. Polit. Econ. (1950).
[7] R.D. Burke, M. Ramezani, Matching recommendation technologies and domains, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 367–386.
[8] M. Stratigi, E. Pitoura, J. Nummenmaa, K. Stefanidis, Sequential group recommendations based on satisfaction and disagreement scores, J. Intell. Inf. Syst. (2021).
[9] E. Triantaphyllou, Multi-Criteria Decision Making Methods: A Comparative Study, Vol. 44, 2000, http://dx.doi.org/10.1007/978-1-4757-3157-6.
[10] S. Amer-Yahia, S.B. Roy, A. Chawlat, G. Das, C. Yu, Group recommendation: Semantics and efficiency, PVLDB 2 (1) (2009) 754–765.
[11] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, M. Shaoping, Fairness-aware group recommendation with pareto-efficiency, in: RecSys, 2017.
[12] T.N. Nguyen, F. Ricci, A. Delic, D. Bridge, Conflict resolution in group decision making: insights from a simulation study, User Model. User-Adapt. Interact. 29 (5) (2019) 895–941.
[13] A. Delic, J. Neidhardt, T.N. Nguyen, F. Ricci, An observational user study for group recommender systems in the tourism domain, Inf. Technol. Tour. (2018).
[14] F.M. Harper, J.A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19.
[15] M. Wan, R. Misra, N. Nakashole, J. McAuley, Fine-grained spoiler detection from large-scale review corpora, 2019, arXiv:1905.13416.
[16] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: WWW, 2016.
[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: An open architecture for collaborative filtering of netnews, in: CSCW, 1994.
[18] C. Desrosiers, G. Karypis, A Comprehensive Survey of Neighborhood-Based Recommendation Methods, Springer US, Boston, MA, 2011, pp. 107–144.
[19] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. Artif. Intell. 2009 (2009).
[20] D. Sacharidis, Top-n group recommendations with fairness, in: SAC, 2019.
[21] J. Masthoff, Group recommender systems: Aggregation, satisfaction and group attributes, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 743–776.
[22] A. Jameson, B. Smyth, Recommendation to Groups, Springer-Verlag, 2007, pp. 596–627.
[23] J.F. McCarthy, T.D. Anagnost, Musicfx: an arbiter of group preferences for computer supported collaborative workouts, in: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, 1998, pp. 363–372.
[24] Z. Yu, X. Zhou, Y. Hao, J. Gu, Tv program recommendation for multiple viewers based on user profile merging, User Model. User-Adapt. Interact. 16 (1) (2006) 63–82.
[25] L. Baltrunas, T. Makcinskas, F. Ricci, Group recommendations with rank aggregation and collaborative filtering, in: RecSys, 2010.
[26] E. Ntoutsi, K. Stefanidis, K. Nørvåg, H.-P. Kriegel, Fast group recommendations by applying user clustering, in: International Conference on Conceptual Modeling, Springer, 2012, pp. 126–140.
[27] M. O'connor, D. Cosley, J.A. Konstan, J. Riedl, Polylens: A recommender system for groups of users, in: ECSCW, 2001.
[28] Q. Yuan, G. Cong, C.-Y. Lin, Com: A generative model for group recommendation, in: KDD, 2014.
[29] D. Cao, X. He, L. Miao, Y. An, C. Yang, R. Hong, Attentive group recommendation, in: SIGIR, 2018.
[30] H. Yin, Q. Wang, K. Zheng, Z. Li, J. Yang, X. Zhou, Social influence-based group representation learning for group recommendation, in: ICDE, 2019.
[31] A. Salehi-Abari, C. Boutilier, Preference-oriented social networks: Group recommendation and inference, in: RecSys, 2015.
[32] L. Vinh Tran, T.-A. Nguyen Pham, Y. Tay, Y. Liu, G. Cong, X. Li, Interact and decide: Medley of sub-attention networks for effective group recommendation, in: SIGIR, 2019.

[33] D. Qin, X. Zhou, L. Chen, G. Huang, Y. Zhang, Dynamic connection-based social group recommendation, in: IEEE TKDE, 2018.
[34] J.K. Kim, H.K. Kim, H.Y. Oh, Y.U. Ryu, A group recommendation system for online communities, Int. J. Inf. Manage. (2010).
[35] M. Kaya, D. Bridge, N. Tintarev, Ensuring fairness in group recommendations by rank-sensitive balancing of relevance, in: RecSys, 2020.
[36] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Comput. Surv. 51 (4) (2018) 66:1–66:36.
[37] C. Cheng, H. Yang, M.R. Lyu, I. King, Where you like to go next: Successive point-of-interest recommendation, in: International Joint Conference on Artificial Intelligence, 2013.
[38] D. Lian, V.W. Zheng, X. Xie, Collaborative filtering meets next check-in location prediction, in: WWW, 2013.
[39] Q. Liu, S. Wu, L. Wang, T. Tan, Predicting the next location: A recurrent model with spatial and temporal contexts, in: AAAI Conference on Artificial Intelligence, 2016.
[40] F. Garcin, C. Dimitrakakis, B. Faltings, Personalized news recommendation with context trees, 2013, CoRR abs/1303.0665.
[41] B. Hidasi, M. Quadrana, A. Karatzoglou, D. Tikk, Parallel recurrent neural network architectures for feature-rich session-based recommendations, in: RecSys, 2016.
[42] N. Hariri, B. Mobasher, R. Burke, Context-aware music recommendation based on latenttopic sequential patterns, in: RecSys, 2012.
[43] D. Jannach, L. Lerche, M. Jugovac, Adaptation and evaluation of recommendations for short-term shopping goals, in: RecSys, 2015.
[44] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: RecSys, 2017.
[45] E. Pitoura, K. Stefanidis, G. Koutrika, Fairness in rankings and recommendations: an overview, VLDB J. 31 (3) (2022) 431–458.
[46] R. Borges, K. Stefanidis, F2VAE: a framework for mitigating user unfairness in recommendation systems, in: SAC, 2022.
[47] R. Borges, K. Stefanidis, On measuring popularity bias in collaborative filtering data, in: EDBT/ICDT Workshops, 2020.
[48] W. Wang, W. Zhang, J. Rao, Z. Qiu, B. Zhang, L. Lin, H. Zha, Group-aware long- and short-term graph representation learning for sequential group recommendation, in: SIGIR, 2020.
[49] M. Stratigi, J. Nummenmaa, E. Pitoura, K. Stefanidis, Fair sequential group recommendations, in: SAC, 2020.
[50] J. Masthoff, Group modeling: Selecting a sequence of television items to suit a group of viewers, User Model. User Adapt. Interact. 14 (1) (2004) 37–85.
[51] A. Piliponyte, F. Ricci, J. Koschwitz, Sequential music recommendations for groups by balancing user satisfaction, in: User Modeling, Adaptation, and Personalization, 2013.
[52] M.M. Afsar, T. Crump, B. Far, Reinforcement learning based recommender systems: A survey, 2021, arXiv:2101.06286.
[53] N. Taghipour, A. Kardan, S.S. Ghidary, Usage-based web recommendations: A reinforcement learning approach, in: RecSys, 2007.
[54] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, Drn: A deep reinforcement learning framework for news recommendation, in: WWW, 2018.
[55] L. Huang, M. Fu, F. Li, H. Qu, Y. Liu, W. Chen, A deep reinforcement learning based long-term recommender system, Knowl.-Based Syst. (2021).
[56] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for list-wise recommendations, 2019, arXiv:1801.00209.
[57] Z. Yuyan, S. Xiayao, L. Yong, A novel movie recommendation system based on deep reinforcement learning with prioritized experience replay, in: ICCT, 2019.
[58] O. Moling, L. Baltrunas, F. Ricci, Optimal radio channel recommendations with explicit and implicit feedback, in: RecSys, 2012.
[59] G. Shani, D. Heckerman, R.I. Brafman, C. Boutilier, An mdp-based recommender system, J. Mach. Learn. Res. 6 (9) (2005).

PUBLICATION IV

Why-Not Questions & Explanations for Collaborative Filtering

Maria Stratigi, Katerina Tzompanaki, and Kostas Stefanidis

DOI: 10.1007/978-3-030-62008-0_21. URL: https://doi.org/10.1007/978-3-030-62008-0%5C_21

Why-Not Questions & Explanations for Collaborative Filtering

Maria Stratigi[1], Katerina Tzompanaki[2], and Kostas Stefanidis[1]( )

[1] Tampere University, Tampere, Finland
{maria.stratigi,konstantinos.stefanidis}@tuni.fi
[2] ETIS Lab, UMR 8051, CY Cergy Paris University, ENSEA, CNRS, 95302 Pontoise, France
aikaterini.tzompanaki@cyu.fr

Abstract. Throughout our digital lives, we are getting recommendations for about almost everything we do, buy or consume. However, it is often the case that recommenders cannot locate the best data items to suggest. To deal with this shortcoming, they provide explanations for the reasons specific items are suggested. In this work, we focus on explanations for items that do *not* appear in the recommendations they way we expect them to, expressed in why-not questions, to aid the system engineer improve the recommender. That is, instead of offering explanations on every item proposed by the system, we allow the developer give feedback about items that were not proposed. We consider here the most traditional category of recommenders, i.e., the collaborative filtering one, and propose ways for providing explanations for why-not questions. We provide a detailed taxonomy of why-not questions on recommenders, and model-specific explanations based on the inherent parameters of the recommender. Finally, we propose an algorithm for producing explanations for the proposed why-not questions.

Keywords: Explanations · Why-not questions · Recommendations · Collaborative filtering · Recommender systems

1 Introduction

Recommendations have been integrated in many of the services available to users in recent times. Although, recommenders try to accurately propose interesting items to users according to their preferences, it is often the case that they cannot locate the best data items to suggest. This can be due to many different reasons. One reason can be the cold start problem, where the system does not have enough information about a user to make accurate predictions. Another cause may be the over-specification on the part of the users. This means that a user has previously expressed a preference for a specific category and the system is unlikely to propose items that belong to a different category. Furthermore, often the systems can be misdirected due to ambiguous information on the users and

Z. Huang et al. (Eds.): WISE 2020, LNCS 12343, pp. 301–315, 2020.
https://doi.org/10.1007/978-3-030-62008-0_21

their preferences. Finally, as a system relies a lot on its hyper-parameters and thresholds, unlucky recommendations may be tight to the system's configuration.

The problem of explaining recommendations is a long-standing problem, which is most regularly approached by introducing explanations along with recommendations (e.g., [4,16,22]). This way, the user or the system's designer gets insights on why an item is suggested. The explanations can then vary on granularity or presentation format based on the final consumer, i.e., the final user of the recommender or the designer of the system. In this work, we expand on the concept of *post-hoc*, model-based explanations [23], i.e., explanations provided after the recommendations have been produced and based on the knowledge of the system, by exploiting the concept of why-not questions. These questions are not about why items were proposed, but why items *were not proposed*. We judge that this kind of questions are necessary for the system engineer, who needs to better understand the system and get hints on how to debug it. For example, assume a system that recommends products to users. If the engineer finds that the products of a specific company are never proposed to a user, he/she may need to understand why, and find the best way(s) to turn the situation around. This could be in the benefit of the diversity of recommendations proposed to the final user, or even for promotional campaigns of the specific company, who does not see their products proposed by the system. On the other side, asking a why-not question may not be such a straightforward task for a final user, who is totally unaware of the context or his/her preferences. However, it can still be applied in the case of a knowledgeable user, who is aware of the context of the recommendations. For instance, a female user of a career development site may wonder why she never gets suggestions for managerial positions. In this case, a why-not explanation would help the user gain trust on the system and promote its gender-fairness. In this work, we assume the system designer as the consumer of the explanations, and leave the case of the final user as a future work.

One could suggest that explaining why a certain item is not proposed is dual to explaining why all the recommended items are proposed. With the standard explanation method, a user has to go over the recommended list and understand the differences between the proposed items and the one(s) expected. This would be a time consuming, or even impossible, task, depending on the user's understanding of the data set and the recommendation model. For this reason, already existing systems that treat the 'why' aspect of explanations cannot trivially explain missing recommendations, especially without the user feedback in the form of a why-not question. By having the system answering why-not questions, this process is streamlined and not strictly dependent on the user knowledge.

In this paper, we consider the traditional paradigm of a user-based collaborative filtering recommendation system for providing explanations to why-not questions. First, we provide a detailed categorization of why-not questions characterized by three main properties: (i) the level of absenteeism that the why-not questions mention (absence or low position in the ranking of a result set), (ii) their granularity (referring to a single result or a set), and (iii) their dependency to existing recommended items. Note, that a why not question may belong to multiple classes. Second, we provide fine-grained and personalised model-based explanations targeted for *system engineers*. The explanations are not dependent on the context of the system (e.g., social, product, PoI recommendation). We distinguish explanations between the *general* ones, based on the general setting of the problem, and the *model-specific* explanations, based on the inherent parameters of the recommendation model. Fourth, we propose an algorithmic method for computing explanations for why-not questions in collaborative filtering. Finally, we conduct a preliminary experimental study that explores the explanations space and motivates their usage by a system designer.

2 Preliminaries and Related Work

For a general setting of our recommender, assume a set of data items I and a set of users U, where each user provides ratings for a subset of I. Specifically, a user $u \in U$ rates an item $i \in I$ with a score s. The subset of users that rated an item i is denoted by U_i, whereas the subset of items rated by a user u, is denoted by I_u. For every item i not rated by a user u, the recommender estimates a relevance score, $p(u, i)$. The items with a high relevance score for u will compose the recommendation list (called also recommended items) for the user.

The literature regarding how to estimate the relevance score of an item for a user is extensive. In this work, we will focus on *collaborative filtering*, a well established recommendation approach that recommends items that users with similar preferences like (e.g., [3,11,13]). Specifically, the collaborative filtering (CF) approach is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. The steps of a CF algorithm to produce a list of recommendations for a user u are: (1) Find the most similar users $Peers_u$ with u by means of a similarity function $sim(u, u')$ between u and every other user u'. (2) Predict a relevance score p for each item not rated by u based on his/her similar users $Peers_u$. (3) Recommend a list R_u with the top-k items with the highest relevance score.

The first step of the CF algorithm is to compute similarities between the users. To measure the similarity $sim(u, u')$ between two users, we exploit their ratings that are available in the recommender. Several metrics appear in the related work for counting similarities between users based on ratings. We employ here the Pearson correlation measure [11], which is fast to compute and performs very well for the case of collaborative filtering. It directly calculates the correlation between two users with a score from -1 for entirely dissimilar users, to 1 for identical users. A user u' is considered similar to u if their similarity is above a threshold th and if they have rated more than $numI$ common items. We further refine the process, by keeping only the $numP$ users with the highest similarity scores. We name these users as *peers* of u, $Peers_u$. In the second step of the algorithm, we use the peers of u to predict a relevance score $p(u, i)$ for any item i that u has not yet rated. To this end, we use the *weighted sum of others ratings* [18]. We only recommend items to u if more than $numPI$ peers have rated them. In this way, we have a more robust understanding of the items' preference by the peers. Additionally, we do not have many *false positives*, by avoiding proposing an item only liked by one (or few) peer(s), while it is unknown to the rest. In the final step, we sort all the items we have predicted a score for and return the k items with the highest score in the list R_u. Furthermore, we denote by $pos_{R_u,i}$ the index of the item i in the list R_u.

Related Work on Explanations in CF. CF explanations are typically provided based on users implicit or explicit feedback (for a survey of explanations in recommenders, refer to [20]). For example, a direct solution is to first find a set of peers for the user in question and then produce a recommendation to this user. The explanation is that the user is similar to the peers, and the peers made good ratings on the recommended item [14]. [9] compares the effectiveness of different display styles for explanations in CF. Specifically, explanations can be displayed as an aggregated histogram of the ratings of the peers, or be displayed as the detailed ratings of the peers. Alternatively, explanations can be provided by telling the user that the recommended item is similar to other items the user liked before, where several highly rated items by the user are shown as explanations [15]. To study the usefulness of explanations in recommender systems, [19] developed a prototype system to study the effect of different types of explanations. In brief, this study shows that providing appropriate explanations can benefit the recommender system over specific goals, like transparency, persuasiveness, trustworthiness and satisfaction. More recently, there exist approaches, e.g., [2,6,21], for generating explanations with methods using matrix or tensor factorization, where the goal is to make latent factors more tangible. From a different perspective, [8] studies the problem of computing minimum subsets of user actions to change the top-ranked recommendations in a counterfactual setup. The concept of why-not questions is used also for probabilistic range queries in [5], either by modifying the original query or by modifying the why-not set. [7] offers a similar framework for why-not questions on reverse top-k queries. To the best of our knowledge, we are the first to define and study the problem of providing explanations based on why not questions in recommender systems.

3 Why-Not Questions

In this paper, we expand on the concept of explanations in recommender systems, by exploiting the concept of why-not questions. These questions are not about why an item is recommended but why an item is not recommended in the expected way. Instead of offering explanations on every item that is proposed by the system, we allow the user to give feedback in the form of questions. For instance, in a movies recommender if the user is not satisfied with the movies list provided by the system, he/she can ask questions like: *Why were there not any comedies recommended?* The system will answer with information based on the system characteristics and the data associated with these items. This paradigm is not yet explored in recommendation systems, while it has been recently explored in other contexts like in explaining query results in relational databases [1], in reverse skyline queries [10], and briefly in machine learning systems [12].

We propose to characterise why-not questions by three main properties: (i) their level of absenteeism, (ii) their granularity, and (iii) their dependency to existing recommended items. The first property is naturally derived from the notion of false negative results, i.e., the items that should have been returned (in a certain position) but are not. The second property goes one step beyond to express groupings of missing items (that can be regarded as false negatives). The third property, corresponds to the need of the system expert to express the fact that an item that is returned (true positive) and an item that is not (false negative) should be encountered together in a result set.

First, we examine why-not questions based on *absenteeism*. In this respect, we further distinguish between (i) *total* absenteeism, and (ii) *position* absenteeism. Question such as *Why not Titanic?* belong to the *total* category, since they are about items that do not appear in the recommendation list, without a specific requirement for the position on which they should appear. Questions that ask about the ranking of items, such as *Why not rank Titanic first?* belong to the *position* category. It is evident thus, that a position absenteeism why-not question can be applied on items that *are* recommended, but still not as highly as expected. Second, we review *Granularity.* Granularity describes the level of detail of the question that is asked, distinguishing between *atomic* cases and *group* cases. In more detail, the user is able to ask questions about specific items (atomic case), such as: *Why not Titanic?*, or about set of items that share a common characteristic (group case), such as: *Why not comedies?* Third, the *Dependency* property describes items that usually appear together in the answers, or should be returned in a specific order. Example questions are *Why there are not any comedies but there are dramas?* This kind of questions fall also in the case of group recommendations, when users expect to find groups of items together. We subsequently define why-not questions in a formal way.

Definition 1. *Let I be a set of items, u a specific user of a recommender system built on I, $R_u \subseteq I$ the set of recommended items for user u by a recommender system. A why-not question is a set of the form*

$$wn = \{(m, pos, d) \mid m \in I \text{ and } pos \in \{1, \ldots, |R_u|\} \text{ and } d \in R_u\}$$

Definition 1 is general enough to cover all the cases that we mentioned before, i.e., absenteeism (m and pos), granularity, and dependency (d). Even though not explicitly apparent, a *granularity* why-not question can be derived by expanding the group to the related items in I. For example, a why-not questions of the style *Why not comedies?* can be represented by $wn = \{(Big, ,), (Zoolander, ,)\}$, given that the system can find these two comedies in its database. Moreover, if the user wants to ask one specific type of a why-not question that does not involve all three parts m, pos, and d, then he/she can leave that part empty. For example, in the case of a total absenteeism why-not question of the style *Why not Titanic?*, the corresponding wn would be $\{(Titanic, ,)\}$. In the next paragraphs we elaborate more on the different types of why-not questions that we consider and how they are expressed using Definition 1.

Next, we specify Definition 1 to express the different properties of why-not questions (absenteeism, granularity, dependency). As the *absenteeism* property is always apparent in a why-not question, we discuss both sub-categories of the *absenteeism* property (total and position), with respect to granularity and dependency. In order to keep the notation from becoming too cumbersome, we will only give the notation for set of items (the group subcategory of the granularity property). This does not affect the formalization of the why-not questions, since both granularity questions can be noted using a set format (an individual item belongs to a set that consists of just one item). For each case, we present an intuitive description, examples in the context of a movie recommendation system, and the formal expression corresponding to that case of why-not question.

- **Total Absenteeism:**
  - *Independent*: The user asks why some items do not exist in the recommendation list.
    Example-Atomic: *Why is there not Titanic?*
    Example-Group: *Why are there not any comedies?*
    Formally, an independent total absenteeism why-not question is:

$$wn_{ti} = \{(m, ,) \mid m \in I \setminus R_u\}$$

  - *Dependent*: The user asks why certain items do not exist while other (that usually appear together) exist.
    Example-Atomic: *Why is there not Titanic while there is Up?*
    Example-Group: *Why not any thrillers when there are action films?*
    Formally, a dependent total absenteeism why-not question has the form:

$$wn_{td} = \{(m, , d) \mid m \in I \setminus R_u \text{ } and \text{ } d \in R_u\}$$

- **Position Absenteeism:**
  - *Independent*: The user can question the ranking of a set.
    Example-Atomic: *Why is Titanic not ranked first?*
    Example-Group: *Why are comedies not in a higher ranking?*
    Formally, an independent position absenteeism why-not question is:

$$wn_{pi} = \{(m, pos,) \mid m \in R_u \text{ } and \text{ } pos_{R_u,m} < pos\}$$

  - *Dependent*: The user asks why certain items do not appear higher in the recommendation list than other recommended items.
    Example-Atomic: *Why not place Titanic before Up?*
    Example-Group: *Why not place comedies before dramas?*
    Formally, an independent position absenteeism why-not question is:

$$wn_{pd} = \{(m, pos, d) \mid m \in I \text{ } and \text{ } pos > pos_{R_u,d} \text{ } and \text{ } d \in R_u\}$$

4 Why-Not Explanations

To answer a why-not question, we seek to provide meaningful explanations to the system designers. By meaningful, we mean the information that is adequate to help the designer understand why the items are not recommended in the expected way, and subsequently use this information in order to repair the system. For this reason, we split the input of the problem to distinct components that can explain - either individually or combined - the why-not question provided by the user. These components are: the input item set, the sets of all and of similar users given the user in question, the set of ranking scores, and the recommender system design (hyperparameters) itself. To accommodate the different sources of error, we define a multi-type structure, called an *explanation*, as follows:

Definition 2. *A why-not explanation for a why-not question on the recommendations of a user u is a set of parameters of the recommender system, responsible for the absence of the missing item(s) from the (specific positions of the) recommendation list.*

We distinguish between *general* explanations, which can appear in any recommender system, and *model-specific* explanations that are based on the inherent parameters of the CF recommendation model. We further describe the general and CF explanations in Sects. 4.1 and 4.2, respectively, while we provide an algorithm to compute them in Sect. 5. We accompany the discussion with examples of why-not questions and respective possible explanations, summarized in Table 1. For clarity, we include in this table a description in natural language for the question and the explanation. The descriptions of the explanations can be regarded as the output of a statistical analysis of the resulting explanations.

4.1 General Explanations

Users, in most cases, ask about an item that does not appear in the recommendation list. So, it is very likely that this item does not exist in the database of the system. The explanation, then, is straightforward; this item is not suggested because it is unknown to the system. Another explanation emerges from the number of returned top-k items. If that number is low, then the missing item may be further down the recommendation list. However, we do not consider that the selected k may be the problem if the item is found at an index greater than $2k$, to promote other potential (model-specific) explanations. Finally, the system may produce the same score for different items. To break the ties, it adopts a specific method, e.g., it will place first the first encountered item in the database. So, when a user poses a why-not question on an item that has been neglected due to the tie-breaking method, the system may designate the tie-breaking method as a culprit. E1-E3 in Table 1 are examples of such explanations.

4.2 CF Explanations

The concept behind CF is that the system suggests items to a user that his/her similar users have liked in the past. This makes all the possible explanations revolving around the user's peers. One scenario is that none of the peers have rated an item. In this case, the item is invisible to the system and cannot be suggested. A similar scenario is for that item to have never been rated before by any user. This again makes the item invisible to the system. Aside from these two explanations, an answer for a "Why not item A?" question is the combination of the results for the three following questions: (i) how many peers have rated it, (ii) what scores they have given it, and (iii) how similar they are to the user. If just one or two peers have rated an item, then the system ignores it, to avoid a false suggestion. If all or most of the peers have given a low score to an item, then the system, in turn, calculates a low score for it. Finally, the similarity that a peer shares with the user is also primordial. If a peer who has a high similarity with the user does not like an item, then this has an impact on that item's final predicted score. E4-E11 in Table 1 are examples of such explanations.

We represent the results of the three aforementioned questions "How many peers have rated it?", "What scores have they given it?" and "How similar are these peers to the user?" as a set of tuples of the form $(peer, score, similarity)$. Each tuple describes a peer who has rated the targeted item and consists of three values: (i) the peer's id, (ii) the score that he/she has given to the item, and (iii) the similarity shared between the peer and the user. If the set is empty then none of the peers have rated this item and we provide explanation E10 (Table 1). If the set consists of only one or two tuples (peers) then it corresponds to explanation E4. To produce the rest of the explanations, we combine into a user-friendly explanation the number of peers that have rated the item, the similarity that these peers share with the user and the scores they have given to the item.

When a user questions the item's ranking in the recommendation list, the system again checks the same information. The system answers questions like: "Why was not item A ranked higher?" by explaining the item's statistics: how many peers have rated the item, if they favored it and how similar these peers are to the user. This type of question is vague, in the sense that the user questions the general ranking of an item without comparing it to another item; the user issued an independent question. So the system treats it as if it was a total absenteeism question. For this reason, explanations E17-E19 are the same as for a total absenteeism why-not question. Another alternative for handling these types of questions is to transform them from independent to dependent by arbitrarily selecting an item in the list. We select this item according to the specifics of the question - higher or lower ranking. For example, a question like "Why was not item A ranked higher?" can be transformed into "Why not place item A before item B?". In this case, the system returns a more detailed explanation as shown in lines E20-E23 in Table 1.

Explanations to dependent why-not questions, such as "Why not item A but item B?" or "Why not place item A before item B?", are more complicated since they involve multiple items, some of which exist and others not, mixing explanations and why-not explanations. We explain the process for the first question. The second is answered in a similar way. First, we decompose the why-not question to two separate queries. The first is a part of the user's question:

Table 1. Examples of why-not questions and explanations in CF.

WN Question	Model	WN Explanation Description	Id
Any why-not question	General/I	Item A does not exists in the database	E1
	General/k	You asked for few items	E2
	General/Tie	Item had the same score as another item	E3
Why not suggest item A?	CF/numPI	Only x (<numPI) of your peers has rated this item	E4
	CF/$\{(peer, s, sim)\}$	x of your peers have given a low score to this item	E5
	CF/$\{(peer, s, sim)\}$	x of your most similar users have given a low score to A	E6
	CF/$\{(peer, s, sim)\}$	x peers like A, but y dislike it	E7
	CF/$\{(peer, s, sim)\}$	All of your peers have given the item a low score	E8
$wn = (A, ,)$	CF/numP	None of your most similar users have rated A, but x with a lower similarity have given it a high/low score	E9
	CF/Peers	None of your peers has rated this item	E10
	CF/S	No one has rated this item	E11
Why not suggest item A but suggest B?	CF/numPI	x of your peers have rated item B but only y (<numPI) has rated item A	E12
	CF/$\{(peer, s, sim)\}$	x of your peers like item B but dislike A	E13
	CF/$\{(peer, s, sim)\}$	x peers like item B and y dislike item A	E14
$wn = (A, , B)$	CF/numP	Your most similar peers have not rated A but have rated B	E15
	CF/Peers	Your peers have rated B but none of them have rated A	E16
Why is not item A ranked higher?	CF/$\{(peer, s, sim)\}$	x of your peers have given a low score to this item	E17
	CF/$\{(peer, s, sim)\}$	x of your most similar peers have given a low score to A	E18
$wn = \{(A, pos_{R_u}A - 1,)\}$	CF/$\{(peer, s, sim)\}$	x peers like A, but y dislike it	E19
Why is not item A higher than B?	CF/$\{(peer, s, sim)\}$	x of your peers like item B but dislike A.	E20
	CF/$\{(peer, s, sim)\}$	x peers like item B and y dislike A	E21
	CF/numP	Your most similar peers have not rated A but have rated B	E22
$wn = \{(A, pos_{R_u}B - 1, B)\}$	CF/Peers	Your peers have rated B but none of them have rated A	E23
Why not suggest comedies?	CF/Peers	None of your peers rated the same movie	E24
	CF/$\{(peer, s, sim)\}$	Your peers dislike comedies	E25
	CF/Peers	None of your peers has rated a comedy	E26
	CF/$\{(peer, s, sim)\}$	Only x of your peers like comedies	E27
$wn = \{(C_1, ,), \ldots, (C_n)\}$	CF/numP	Your most similar peers do not like comedies but x of your least similar do	E28

"Why not item A". The second query we make is "Why not item B"[1]. Intuitively, since the system has promoted item B to the user instead of A (either by not even suggesting A for *total* why-not questions or with a better ranking in the recommendation list for *position* why-not questions), the results of the questions "How many peers have rated it?", "What scores have they given it?" and "How similar are these peers to the user?" have higher values than the results for A. Then, we combine the answers of these two why-not questions. For example, see E12-E16 in Table 1. We choose to explain the existence of item B as a why-not explanation, because it allows us to combine the results of the two questions more effectively than if we used a generic explanation method, such as [9].

When the user formulates a *group* why-not question, for example "Why not more comedies?", the explanation that the system provides is a union of all the answers that it would have provided for individual items. For each item in the same category as the one the user asked about and a peer has preferred in the past, we formulate a why-not question. Since this can become very cumbersome for the user to consume, one can summarize the results into a user-friendly output. For instance, "Your peers like comedies, but they have not liked the

[1] An alternative here could be to employ a solution for explaining recommendations.

same one", meaning that the peers have shown some preference for comedies, but each peer has rated a different movie. This explains why none of them were suggested. To reduce the number of questions we issue to the system, we can take into consideration the items that the peers have shown a great preference for. These items are the most important for us, since they have the highest probability to be suggested. For example, in a system where the ratings are in the range from 1 to 5, we can only consider the items that have a rating higher than the average 2.5. We provide more explanations for a group why-not question in lines E24-E28 of Table 1.

Algorithm 1: *WNCF*

```
Input: item set $I$, user set $U$, user $u$, why-not question $\{(i, , )\}$, rating scores $S$,
       recommendation list $R_u$ for user $u$, threshold $numPI$, threshold
       $numP$,peers of $u$ $Peers$, relevance score function $p(u, i)$
Output: $e$, explanation
1  if $i \notin I$ then
2      $e$.add('I');
3  else if $\exists i' : p(u, i)=p(u, i')$ and $pos_{R_{i'}} \leq k$ then
4      $e$.add('Tie');
5  else if $i$ in the $2k$ first entries of expanded $R$ then
6      $e$.add('k');
7  else if $i$ has no ratings in $S$ then
8      $e$.add('S');
9  else if at least one peer of $u$ has rated $i$ then
10     for $peer \in Peers$ do
11         if $peer$ has rated $i$ then
12             $e$.add(($peer$, $s(peer, i)$,$sim(u, peer)$));
13     if less than $numPI$ most similar peers of $u$ have rated $i$ then
14         $e$.add('numP');
15     if less than $numPI$ peers of $u$ have rated $i$ then
16         $e$.add('numPI');
17 else
18     for $u'$ user in $U$ do
19         if $u'$ has rated $i$ then
20             $e$.add(($u'$, $s(u', i)$,-));
21     $e$.add('Peers');
22 return $e$;
```

5 *WNCF* Algorithm

In this section, we introduce *WNCF* (standing for *Why-Not in Collaborative Filtering*), an algorithm for the computation of why-not explanations for the CF model (Algorithm 1). *WNCF* addresses total absenteeism for atomic granularity independent why-not questions. The extension of this algorithm to explain other types of why-not questions, as discussed in Sect, 4, is trivial for some cases, e.g., group independent why-not questions, but not for all. We postpone the extensions to future work.

As mentioned in Sect. 4, we first check if a general explanation can be provided; if not, we proceed with the model-specific explanations modeled in tuples representing the peers who have rated i, along with information on their rating on i and the similarity to user u. If such peers do not exist, we provide explanations based on other users who have rated i.

In more detail, *WNCF* receives as input the item set, user set, the ranking scores, and the threshold values $numPI$ and $numP$ of the CF system, as well as the why-not question for a user u. We also consider known the peers of u and the recommendation list calculated for u, as well as the relevance score function. Line 1 checks if the specific item exists in the database. If it does not, we return the explanation code I, to indicate that the source of error is the input data set. Line 3 checks if the item shares the same relevance score with another item that appears in the list. In this case, we return the explanation code Tie, to indicate that the source of error is the tie breaking method. Line 5 checks if the item appears between the kth and $2k$th entry. In this case, we return the explanation code k, to indicate the the k maybe too low. Line 7 checks if any user in the system has rated i. If none of them did, then we return the explanation code S, to indicate that there are not rating scores for i.

Lines 9–16 check the peers of the user. For every peer who has rated i, we report the score he/she has given, as well as the similarity he/she shares with u (Line 12). Then, we check the $numP$ most similar peers of the user (Line 13). If less than $numPI$ of them have rated the item, we return the code ‘numP’ to express that there are not enough most similar peers who have rated the item. Subsequently, we check the rest of the peers and if there were not at least $numPI$ peers who have rated i (Line 15), then we return the explanation code ‘numPI’. This indicates that from all the peers of user u less than $numPI$ peers have rated this item. Finally, if none of the peers has rated this item (Line 17) we return the explanation code $Peers$ to indicate that there are no peers who have rated the item. We also return information about the users (non-peers) who have rated the item and their scores.

Overall, our rational for returning the extra information on the users (peers or not), their ratings for the item in question and their similarity to the user u for CF systems, is two-fold. First, we can compute statistics that can be easily consumed by the developers (in their raw format or as visualisations) and help them understand more about the setting. Second, we can use this information as input to a repair mechanism, which would propose changes to the system so as to make the missing item appear in the list.

6 Experiments

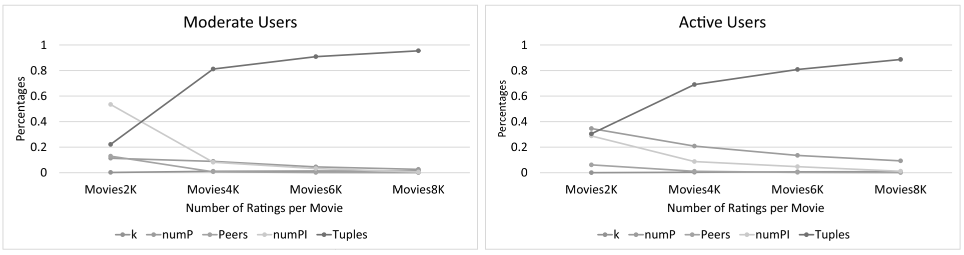
We study *WNCF* with respect to different parameters, namely the characteristics of the users for whom we pose the why-not question, and the popularity of the missing movie. Moreover, we perform an experiment that shows the next step that a developer can take, after he/she receives a *WNCF* explanation.

Experimental Setup. In the experiments we used the MovieLens 20 Million Ratings[2] that consists of 27.278 movies and 138.493 users. To study the behavior of the algorithm for different types of users we randomly selected 100 users that have rated a few items (45 to 55), called *Moderate Users*, and 100 users that have rated many items (145 to 155), called *Active Users*. To experiment with the characteristics of the movies that comprise our why-not questions we used movies of varying popularity. We randomly selected 4 sets of 100 movies that have 2K (least popular), 4K, 6K and 8K (most popular) ratings, respectively. We denote these sets as Movies2K, Movies4K, Movies6K, and Movies8K. We ensured that the movies selected are not in the recommendation lists of the users, in order to be able to run why-not questions with them. To find the peers of a user, we used the Pearson Correlation with a threshold of 0.8 (th), while to predict a score for an unrated item we utilized 100 ($numP$) peers. An item cannot be considered for addition in the recommendation least, unless 3 ($numPI$) or more peers of the user have given it a rating. We report to the user the top-10 movies with the highest predicted scores.

Explanations Study. First, we define a total absenteeism why-not question for each item in the varying popularity movie sets, for moderate and active users (Fig. 1(a) and (b) respectively). Then, we run *WNCF* for each user and why-not question, and we calculate the percentage of occurrences of each explanation depending on the different parameters as they appear in the different segments of Algorithm 1. The k explanation indicates that the item in question was further down in the list that was provided to the user. The $numP$ explanation means that we should augment the $numP$ threshold to be able to find enough most similar peers that have rated this specific item. The *Peers* explanation occurs when none of the peers of the user has rated an item. Finally, by *Tuples* we denote explanations comprised by information on the peers of the user, calculated in Algorithm 1 line 12, and when the conditions in lines 13 and 15 are false.

Let us further analyze the result of the experiment in Fig. 1. When we use the more popular movies, more of them are in the k range, almost all of them are rated by a peer of the user and most of them are rated by a top peer. This is most evident when we compare the results for Movies2K and Movies8K. For Movies2K, less than 20% of the movies could be explained by the information provided on the peers in the moderate case, while for the Movies8K more than 95% of explanations were composed by the peers. Additionally, in the Movies2K more than half of them were not rated by any of the user's peers, while in the

[2] https://grouplens.org/datasets/movielens.



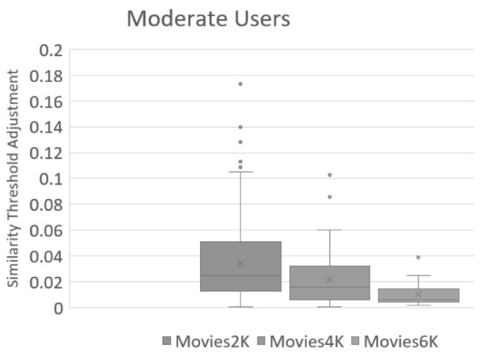
(a) (b)

Fig. 1. Explanations for varying popularity of missing movies for (a) moderate, and (b) active users.

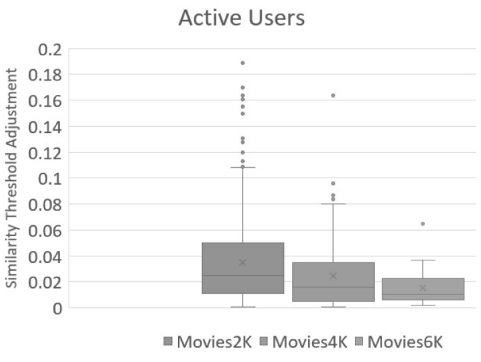
Movies8K this number has dropped down to almost zero. We can observe similar numbers for the Active Users for the Movies2K and Movies8K movie sets. This is a self-evident result since the more popular movies have a higher chance to be rated by the peers of a user. This is further corroborated by the *Tuples* values. For the Movies2K, it has low values, since the movies are not that popular. With each subsequent movies set, as the popularity of the movies rises, so does the the values of the *Tuples* variable. The most popular movies are more likely to have been voted by a top peer.

When comparing the results for the two sets of users, the Active Users have more explanations about the top peers not having rated an item (*numP*) than the Moderate Users. This is because the more ratings a user has given, the more similar other users he/she has. Since the number of peers we use is a constant variable and not a percentage, there is a higher chance the selected users have not rated the movies questioned. At the same time, we can see that the number of movies that were not rated by any of peers is lower than that of the Moderate Users. This is again because Active Users have a higher number of peers.

To demonstrate how the developer can proceed when he/she has acquired one explanation, we considered the case of *Peers* explanations. We took all the movies that were not rated by any peer (corresponding to *Peers* explanations), and we examined all the users in the system in order to find the new threshold needed in the similarity function, so as the recommender to be able to calculate a preference score for that item for the considered users. Then, we calculated the difference between the threshold we used originally (in our experiments, 0.8) and the new calculated threshold. Figures 2a and 2b show the results for the Moderate and Active users, respectively. In both experiments, we excluded the Movies8K set because the number of movies that were not rated by any of the peers is very small (less than 5 in both user sets). In both cases the adjustments needed in the similarity threshold are small. The average values (denoted with x in the figures) is below 0.04. While the Active Users have more outliers (dots in the figures) their ranges are similar to those in the Moderate user set. Finally, the median value (line inside the box) is comparable for both user sets across all movie sets. Thus, we see that with the provided explanation the developer can directly explore the right direction for debugging his system.



(a)



(b)

Fig. 2. The similarity threshold (*th*) adjustment needed for the recommender to be able to calculate a preference score for the missing items corresponding to a *Peers* explanation for (a) moderate, and (b) active users.

7 Summary

In this work, we pay special attention on transparency provided via explanations in recommender systems. We exploit the concept of why-not questions, allowing the user to give feedback in the form of questions about why items are not proposed in the expected way. We consider the collaborative filtering approach, and propose ways for providing explanations for why-not questions. We provide a detailed taxonomy of why-not questions with respect to three main properties: (i) the level of absenteeism that the why-not questions mentions (absence or low position in the ranking of a result set), (ii) their granularity (referring to a single result or a group), and (iii) their dependency to existing recommended items. An explanation for a why-not question is meant to inform the user about the possible sources of error linked to the why-not question. We distinguish explanations between general ones, i.e., explanations that are independent to the recommendation model used, and model-specific ones, based on the inherent parameters of CF. Finally, we provide an algorithm for computing why-not explanations in CF systems. Clearly, there are many directions for future work, including proposing explanations for content-based, hybrid and sequential [17] recommendation models, as well as efficient algorithms and implementations in specific contexts. Furthermore, we target the automatic refinement of the recommendations computed for the users, by exploiting the defined explanations.

References

1. Bidoit, N., Herschel, M., Tzompanaki, K.: Immutably answering why-not questions for equivalent conjunctive queries. In: TaPP (2014)
2. Borges, R., Stefanidis, K.: On measuring popularity bias in collaborative filtering data. In: EDBT/ICDT Workshops (2020)
3. Breese, J.S., Heckerman, D., Kadie, C.M.: Empirical analysis of predictive algorithms for collaborative filtering. In: UAI (1998)
4. Chang, S., Harper, F.M., Terveen, L.G.: Crowd-based personalized natural language explanations for recommendations. In: RecSys (2016)
5. Chen, L., Gao, Y., Wang, K., Jensen, C.S., Chen, G.: Answering why-not questions on metric probabilistic range queries. In: ICDE (2016)
6. Chen, X., Qin, Z., Zhang, Y., Xu, T.: Learning to rank features for recommendation over multiple categories. In: ACM SIGIR (2016)
7. Gao, Y., Liu, Q., Chen, G., Zheng, B., Zhou, L.: Answering why-not questions on reverse top-k queries. Proc. VLDB Endow. **8**(7), 738–749 (2015)
8. Ghazimatin, A., Balalau, O.D., Roy, R.S., Weikum, G.: PRINCE: provider-side interpretability with counterfactual explanations in recommender systems. In: WSDM (2020)
9. Herlocker, J.L., Konstan, J.A., Riedl, J.: Explaining collaborative filtering recommendations. In: CSCW (2000)
10. Islam, M.S., Zhou, R., Liu, C.: On answering why-not questions in reverse skyline queries. In: ICDE (2013)
11. Konstan, J.A., Miller, B.N., Maltz, D.A., Herlocker, J.L., Gordon, L.R., Riedl, J.: Grouplens: applying collaborative filtering to usenet news. Commun. ACM **40**(3), 77–87 (1997)
12. Lim, B.Y., Dey, A.K., Avrahami, D.: Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: CHI (2009)
13. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.: Strength lies in differences: Diversifying friends for recommendations through subspace clustering. In: CIKM (2014)
14. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: CSCW (1994)
15. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW (2001)
16. Stefanidis, K., Ntoutsi, E., Petropoulos, M., Nørvåg, K., Kriegel, H.: A framework for modeling, computing and presenting time-aware recommendations. Trans. Large-Scale Data- Knowl.-Centered Syst. **10**, 146–172 (2013)
17. Stratigi, M., Nummenmaa, J., Pitoura, E., Stefanidis, K.: Fair sequential group recommendations. In: SAC (2020)
18. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. In: Advances in Artificial Intelligence (2009)
19. Tintarev, N.: Explanations of recommendations. In: RecSys (2007)
20. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: ICDE (2007)
21. Wang, N., Wang, H., Jia, Y., Yin, Y.: Explainable recommendation via multi-task learning in opinionated text data. In: ACM SIGIR (2018)
22. Yu, C., Lakshmanan, L.V.S., Amer-Yahia, S.: Recommendation diversification using explanations. In: ICDE (2009)
23. Zhang, Y., Chen, X.: Explainable recommendation: a survey and new perspectives. Found. Trends Inf. Retrieval **14**(1), 1–101 (2020)

PUBLICATION V

Multidimensional Group Recommendations in the Health Domain

Maria Stratigi, Haridimos Kondylakis, and Kostas Stefanidis

Algorithms 13.3 (2020), p. 54
DOI: 10.3390/a13030054. URL: https://doi.org/10.3390/a13030054

algorithms

Article

Multidimensional Group Recommendations in the Health Domain

Maria Stratigi [1,*], Haridimos Kondylakis [2] and Kostas Stefanidis [1]

1 Faculty of Information Technology and Communication Sciences, Tampere University, 33100 Tampere, Finland; konstantinos.stefanidis@tuni.fi
2 Institute of Computer Science (ICS), Foundation of Research & Technology-Hellas (FORTH), 70013 Iraklio, Greece; kondylak@ics.forth.gr
* Correspondence: maria.stratigi@tuni.fi

Received: 6 February 2020; Accepted: 27 February 2020; Published: 28 February 2020

Abstract: Providing useful resources to patients is essential in achieving the vision of participatory medicine. However, the problem of identifying pertinent content for a group of patients is even more difficult than identifying information for just one. Nevertheless, studies suggest that the group dynamics-based principles of behavior change have a positive effect on the patients' welfare. Along these lines, in this paper, we present a multidimensional recommendation model in the health domain using collaborative filtering. We propose a novel semantic similarity function between users, going beyond patient medical problems, considering additional dimensions such as the education level, the health literacy, and the psycho-emotional status of the patients. Exploiting those dimensions, we are interested in providing recommendations that are both high relevant and fair to groups of patients. Consequently, we introduce the notion of fairness and we present a new aggregation method, accumulating preference scores. We experimentally show that our approach can perform better recommendations to small group of patients for useful information documents.

Keywords: recommendations; group recommendations; semantic similarity; group aggregation

1. Introduction

Medicine is undergoing a revolution that is transforming the nature of healthcare from reactive to preventive. These changes came to pass due to new approaches to disease, which focus on integrated diagnosis, treatment, and prevention of disease in individuals. One of the major challenges to this path is the amount and the quality of information that is available online [1] considering that health information is one of the most popular research field on the Web. Furthermore, there is a significant increase to the number of people who search online for health and medical information. In the United States, estimations show that ~80% percent of all adults have searched the Web for health information, whereas in 2006, 23% of the Europeans were utilizing the Internet to be informed about their health problems [2]. However, despite the increase in those numbers, it is very hard for a patient to accurately judge how relevant the information is to their own health issues and additionally if the source of this information is reliable.

A healthcare provider that is responsible for providing reliable sources to patients may be an optimal solution for this problem [1]. This guided solution leads to patient empowerment, meaning that a patient receives information from accurate sources, which increases the understanding of their problems and their way of thinking about them. Accordingly, the patients depend less on the doctors for the appropriate information. Additionally, patients feel autonomous and more confident about the management of their disease [3]. Most primary care providers have their patients' health background and interests in paper, electronic, or mental records. This helps them determine what information would be the most constructive for each individual patient. However, the amount of time that a health provider can dedicate to each patient has greatly declined. Consequently, they have an even more difficult task in guiding their patients.

Aside from the guidelines provided by health providers, another support for the patients is their social circle. The use of group dynamics-based principles of behavior change have been shown to be highly effective in enhancing social support, e.g., through promoting group cohesion in physical activity [4] and in reducing smoking relapse [5]. Especially for cancer, the latest studies [6] suggest that group therapy improves the well-being of cancer patients because of enhanced discussion and social support. In these counseling meetings, the patient is directed towards the most informative and reliable sources on the Web. However, the effort of locating pertinent information for a group of participants is far greater than identifying information for just one patient.

This motivates us to concentrate our efforts on recommending to a group of patients relevant and interesting health documents that were selected by health professionals. We utilize the collaborative filtering (CF) recommendation model for this task. Our motivation for this work is to offer to a caregiver that is in charge for a group of patients a recommendation list that consists of health documents that are relevant to the group members. The relevance of the recommended documents is calculated based on the patients' current health profiles. In addition, we would like to identify information that is equally fair to all members, meaning no member in the group is unsatisfied.

We incorporate fairness during the aggregation phase of our recommendation model. To produce group recommendations, one must first produce recommendation lists for each group member and then aggregate those into one list that is then reported back to the group. There are many methods to ensure that the aggregation is done fairly [7,8]. Intuitively, to achieve fair recommendation we consider that all the group members are equal to each other. Therefore, the group score for an item i is the average score across all the group members' preference scores for that item. Such an approach, however, can easily ignore the opinion of the minority. For example, in a group that consists of three patients, if for all items two of the members have high relevance score, but the third is low, then the opinion of the third member is overshadowed by the other two. To counter such a drawback, we propose a new aggregation method that is done in phases. In each phase, we select one item to include in the group recommendation list. At the beginning of each phase, if there is a member that is not as satisfied as the rest of the group, we select an item based on two criteria. First, it has to have high relevance for that user, and second, it is the best one available for the rest of the group.

As we have already mentioned, to generate these recommendations we use the collaborative filtering method. The basic principle of CF is to find similarities between users. Given a target user, we locate other similar users, who are often called peers or neighbors, and utilizing the ratings that peers have given, we estimate relevance scores for the items that the target user has not yet rated. In our work, to calculate similarities between two patients, we consider their health profiles. The information that is included in these health profiles is the following; the ratings that the patient has given to health documents and the health problems they have. Additionally, each patient has been questioned about their education level and their health literacy—meaning to what degree they are able to understand basic information and services related to the health domain. Furthermore, they are periodically questioned about their psycho-emotional status. Specifically, they regularly fill in a questionnaire about their anxiety levels and cognitive closure, meaning the patient's need when faced with a decision, to have an answer instead of continued uncertainty. We propose a new similarity measure that combines all these different sources of information to different degrees to find similarities between patients across all dimensions.

In the past, we have proposed a semantic similarity function that takes into account the patients medical profiles, showing its superiority over a traditional measure [9,10] in group recommendations in the health domain. In addition, we have focused on the notion of fairness [11], devising an aggregation method for ensuring that if the group recommendation list provides a high relevant document for a patient, then that patient may be tolerant of the existence of documents that are not relevant to him/her.

However, although usually health professionals target closely related health problems, the education level, health literacy level, and psycho-emotional status of the group are of high importance, as the content that the health professional should recommend, should be based on the aforementioned axes. To this direction we further extend the dimensions considered for finding similar users and we introduce a new aggregation method called *AccScores*, outperforming existing ones.

More specifically, the contributions of our work are the following.

1. We demonstrate a multidimensional group recommendation model in the health domain, using collaborative filtering.
2. We propose a novel semantic similarity function that takes into account, in addition to the patients medical problems, the education, the health literacy and the psycho-emotional status of the patients, showing its superiority over a traditional measure.
3. We introduce a new aggregation method accumulating preference scores, called *AccScores*, showing that it dominates other aggregation methods and is able to produce fair recommendations to small groups of patients.
4. We experimentally show the value of our approach, introducing the first synthetic dataset with such information for benchmarking works in the area.

This paper significantly extends our previous work in [11], by introducing two new similarity measures and a way to combine the different similarities functions into one. Furthermore, we introduce a new aggregation method and we present the relevant experiments. To our knowledge, this is the first work in group recommendations in the health domain considering multiple dimensions for increasing the quality of the proposed recommendations. The requirements for generating such a tool originally came from the iManageCancer [12] and the BOUNCE [13] H2020 EU research projects.

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 focuses on identifying similarities between users and on how to produce single user recommendations. Section 4 focuses on the group recommendations model, and Section 5 presents the synthetic dataset constructed for evaluation. Finally, Section 6 presents experimental evaluation, and Section 7 concludes the paper.

2. Related Work

Typically, recommendation approaches [14] are distinguished between content-based, which recommends items similar to those the user previously preferred (see, e.g., [15]), and collaborative filtering, which recommend items that users with similar preferences like (see, e.g., [16]). Nowadays, recommendations have more broad applications [17], beyond products, like links (friends) recommendations [18], query recommendations [19], open source software recommendations [20], diverse venue recommendations [21], sequential recommendations [22,23], or even recommendations for evolution measures [24,25].

Although traditional research on recommender systems has almost exclusively focused on providing recommendations to single users, there exist many cases where the system needs to suggest items to groups of users [26,27]. As an example consider a group of friends deciding to dine at a restaurant. Typically, for producing group recommendations, we first compute recommendations for each group member separately, and then employ an aggregation strategy across them to compile the group recommendations (see, e.g., [28,29]). Various aggregation strategies can be applied to find a consensus between users for particular items, by minimizing, for instance, the disagreements between the group members. More recently, the authors of [30] analyze the problem of recommending sets of items to groups incorporating factors, like user impact, viability, and fairness.

Recommendations in the Health Domain

Nowadays, patients turn towards the Web to inform themselves about their diseases and their possible treatment. This suffers from two main problems. First, the information found on the Web is not always accurate, and second, it is very diverse. To face these problems a personalized recommender would allow the users to have a seamless, secure, and consistent bidirectional linking of clinical research and clinical care systems, and thus empowering the patients to extract the relevant data out of the overwhelming large amounts of heterogeneous data and treatment information. The authors of [31] portray the requirements that a Health Recommender System (HRS) needs to fulfill, whereas the authors of [32] analyze common pitfalls of such systems. For a recent survey for recommender systems for health promotion, the interested reader is forwarded to [33].

In this line of work, there have been already developed many recommendation systems focusing on citizen's wellbeing. For example, the authors of [34,35] propose web-based recommender systems that provides individualized nutritional recommendations according to the user's health profile defined, by following the main guidelines furnished by a medical specialist, whereas the authors of [36] suggest messages relevant to the user to support the smoking cessation process. The work in [37] is a recommender system proposing physical activities using only user's history and employing machine learning, whereas for chronic conditions, other works focus on integrating recommender systems with electronic health records [38,39], proposing the best course of treatment. Other approaches adapt past recommendations to the current state of the user for Diabetes patients [40] or propose context-aware recommendation methods [41] to establish personalized healthcare services. However, all these works use techniques that are principally found in pure group recommendations systems for composing the group recommendation list. However, we have tailored our recommendations for the health domain, exploiting the semantically annotated PHR profile of the users. This directly allows us to endorse documents that are relevant to a user not only on the level of appreciation (meaning the ratings that each item has gained), but also on the level of his personal health profile (we recommend items relevant to him because of related health artifacts). Furthermore, by introducing the concept of fairness in our approach, we make sure that the output of the group recommendation process, remains fair and unbiased towards all group members. This is particularly important in our domain, where we explicitly want all members of the group to be satisfied.

More similar works to our approach are [42–46]. In [42], the authors combine two health information recommendation services—a collaborative filtering and a physiological indicator-based recommender—providing to the users useful health information. The authors of [43,44] present a tool aiming to empower patients to extract relevant data out of the overwhelmingly large amounts of heterogeneous data and treatment information, by semantically annotating both the patient profiles and the past user queries. From a different perspective, the authors of [45] decouples users and items, considering properties related to users and items, based on which a collaborative filtering model is defined. On the other hand, the authors of [46] focus on helping help health providers acquire new knowledge in real-time. However, even in those works, notions like group recommendations and fairness are not considered, nor interesting profile dimensions like the educational level, the health literacy, and the psychoemotional status.

For groups there have been only a small amount of works. The authors of [47] focus on recommending video content in group-based reminiscence therapy. Besides this work, in our previous line of work, we focused on group recommendations in the health domain [9,10] by proposing a semantic similarity function that takes into account the patients medical profiles, showing its superiority over a traditional measure in group recommendations, and by introducing the notion of fairness [11], paving the way for our contribution in this paper. Nevertheless, we are not aware of any other work in the area considering dimensions like the educational level, the health literacy, and the psychoemotional status of the patients for recommending high-quality information.

3. Single User Recommendations

Assume a set of documents I and a set of patients U in a health-related recommender system. Each patient is associated with a personal profile that contains the user's personal health information. Each user is able to score documents that they have read in the past. This set of ratings is also contained in the user's profile.

For the documents that a user has not seen previously, the recommender estimates a relevance score $relevance(u, i)$, $u \in U$, $i \in I$. For computing relevance scores, in this line of work, we apply the collaborative filtering approach. That is, given a user, we first look for similar users/patients employing a *similarity function* that evaluates their proximity (Section 3.2). Then, we compute the documents relevance scores using the most similar users to the user in question (Section 3.3). In this paper, in addition to traditional similarity functions, we exploit the patient profiles for finding similarities, targeting at improving the quality of the recommendations.

3.1. User Profiles

To take advantage user profile information, we need as a first step to be able to record it. For this reason, besides capturing patient problems, specific short validated questionnaires (i.e., the ALGA-C questionnaire [48]) have been employed that are being answered by the members of a group. All information obtained is then modeled and stored by exploiting an ontology. The answers of the questionnaires are then used to automatically compute particular values that are stored in the patient profiles, regarding key profile areas. Among others, numerical scores (1 to 5) exists for health literacy level, educational level, cognitive closure, and anxiety that we further use for providing recommendations. *Health literacy* is the degree to which individuals have the ability to obtain, process, and understand basic information and services related to the health domain, needed to make appropriate health decisions [49]. Although initially the term was related to the individual *educational level*, it is has now been acknowledged as an inconsistent indicator of skill level [50] and, as such, we believe it should be captured individually. Cognitive closure, on the other hand, characterizes the extent to which a person, faced with a decision, prefers any answer in lieu of continued uncertainty [51]. Cognitive closure and anxiety have been related with more rapid and lower quality of decision-making and as such different type of information should be recommended to those patients.

Besides user profiling, the documents also need to have information regarding the target population concerning the aforementioned dimensions. As such, all documents entered by the caregivers are annotated with numbers regarding target population health literacy and education level. In addition, the documents are automatically annotated using ICD-10 (http://www.icd10data.com/) ontology, and all annotations are stored into the document corpus.

Concerning the rating dataset the patient, $u \in U$ might rate a document $i \in I$ with a score $r(u, i)$, in the range of $[1, 5]$. Commonly, patients give ratings only for a few documents, whereas, concurrently, the cardinality of I is high. We denote the subset of patients that rated a document $i \in I$ as $U(i)$, and the subset of documents rated by a user $u \in U$ as $I(u)$.

3.2. User Similarities

The information that is available to us to find similarities between users is diverse. First, we have the ratings that each user has given to documents. Second, we can utilize the users' personal information; their health problems, health literacy, and education levels; as well as their anxiety and cognitive closure scores. Because the knowledge that we gain from each source is distinct, we can define four different similarity functions. To better utilize all of our data, the final similarity score between two users will be the combination of the similarity scores from these four methods.

3.2.1. Similarity Based on Ratings

We assume that two patients have similar interests, and in turn are similar, if they gave similar ratings to the documents of the recommender. We employ here the Pearson correlation measure [16], which is fast to compute and performs very well in the case of collaborative filtering. It directly calculates the correlation between two users with a score from −1 for entirely dissimilar users, to 1 for identical users.

$$RatS(u,u') = \frac{\sum_{i\in X}(r(u,i)-\mu_u)(r(u',i)-\mu_{u'})}{\sqrt{\sum_{i\in X}(r(u,i)-\mu_u)^2}\sqrt{\sum_{i\in X}(r(u',i)-\mu_{u'})^2}} \tag{1}$$

where $X = I(u) \cap I(u')$, μ_u denotes the mean of the ratings in $I(u)$.

3.2.2. Similarity Based on Health Information

It is quite common in health-related informatics to consider people as similar if they have similar health problems, which in turn leads to similar consumption of health documents. In this work, we use the International Statistical Classification of Diseases and Related Health Problems (ICD10), which is a standard medical classification list maintained by the World Health Organization, to keep track of and recognize similarities between health problems and users. We describe ICD10 as a tree, with health problems as its nodes. We use the 2017 version of ICD10, which includes four levels in tree representation, plus one for the root level. Because of the structure of the taxonomy (acyclic), there is only one path that connects two individual nodes. Another characteristic of the structure is that sibling nodes that appear at lower levels have greater similarity than siblings in the upper levels.

Table 1 presents an example of four pairs of sibling nodes from the ICD10 ontology, with their code id, their description, and the level they belong to. From their descriptions, we can identify that the siblings that reside in the forth level share a far greater similarity than the ones in the first level. Because of this discrepancy of the similarity of the health problems at different levels, we assign different weights to nodes taking into account their level. These weights will allow us manage differently sibling nodes at various levels. Intuitively, the goal is to have sibling nodes in the higher levels with greater similarity than those in the lower levels.

Table 1. An instance of the ICD10 ontology.

Code ID	Description	Level
S27	Injury of other and unspecified intrathoracic organs	1
S29	Other and unspecified injuries of thorax	1
S27.3	Other injury of bronchus, unilateral	2
S27.4	Injury of bronchus	2
S27.43	Laceration of bronchus	3
S27.49	Other injury of bronchus	3
S27.491	Other injury of bronchus, unilateral	4
S27.492	Other injury of bronchus, bilateral	4

Definition 1 (Weight). *Let A be a node in the ontology tree. Then,*

$$weight(A) = w * 2^{maxLevel-level(A)} \tag{2}$$

where w is a constant, $maxLevel$ is the maximum level of the tree, and $level(A)$ is a function that returns the level of each node.

Moreover, assume that $anc(A)$ is the direct ancestor of A. Intuitively, we need a formula that not only takes into account the distance between two nodes, but also the level that those nodes belong. To achieve that, we make use of the notion of the lowest common ancestor (LCA).

Definition 2 (LCA). *Let T be a tree. The lowest common ancestor LCA(A,B) of two nodes A and B in T is the lowest node in T that has both A and B as descendants, where each node can be a descendant of itself.*

Then, for counting the distance between A and B, we calculate their distance from $LCA(A,B)$. For doing so, we identify first the path that connects A (and B, respectively) with $LCA(A,B)$.

Definition 3 (Path)**.** *Let T be a tree, and A and B two nodes in T, with $LCA(A, B) = C$. $path(A, C)$ returns a set of nodes including A, its direct ancestor $anc(A)$, its direct ancestor $anc(anc(A))$, and so on, until we reach C, without including C in the set.*

The distance between A and C is computed as the summation the weight of each node in the path:

$$dist(A, C) = \sum_{n \in path(A,C)} weight(n) \tag{3}$$

Overall, for computing the similarity between two nodes A and B, we use the following formula.

Definition 4 (simN)**.** *Let T be a tree, and A and B two nodes in T, with $LCA(A, B) = C$. Then,*

$$simN(A, B) = 1 - \frac{dist(A, C) + dist(B, C)}{maxPath * 2} \tag{4}$$

Note that we divide the sum of the two distances with $maxPath * 2$, to normalize the overall similarity, so that the function $simN$, returns a value in the range of [0,1]. We define $maxPath$ as follows.

Definition 5 (maxPath)**.** *Let T be a tree, and A and B two nodes in T, with A being a node in the highest level and B the root. Then,*

$$maxPath = dist(A, B) \tag{5}$$

Figure 1 presents a snippet of the ICD10 ontology tree, where each node is associated with a weight (in this example, $w = 0.1$). The root has not been assigned a weight, because when calculating the path that connects a node with its ancestor, we do not include the actual ancestor in the path. Table 2 presents various similarities between nodes from Figure 1.

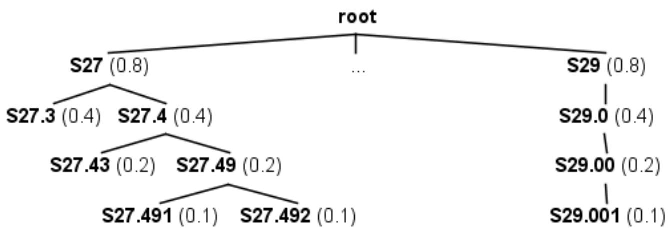


Figure 1. A snippet of the ontology tree along with the assigned weights in parenthesis.

Table 2. Examples of similarities between nodes using Figure 1.

Node A	Node B	LCA(A,B)	simN(A,B)
S27.43	S27.49	S27.4	$1 - (0.2 + 0.2/3) = 0.87$
S27	S29	root	$1 - (0.8 + 0.8/3) = 0.47$
S27.492	S27.49	S27.49	$1 - (0 + 0.1/3) = 0.97$
S27.3	S27.49	S27	$1 - (0.4 + 0.6/3) = 0.67$
S27.492	S29.001	root	$1 - (1.5 + 1.5/3) = 0$
S27.491	S27.492	S27.49	$1 - (0.1 + 0.1/3) = 0.93$

Overall Semantic Similarity Between Two Users

Using the measures described above, we can compute the similarity between two health problems. However, a patient typically has more than one health problem in his/her profile.

Let $Problems(u)$ be the set of health problems of a patient $u \in U$. Given two patients, u and u', their overall similarity is calculated by considering all possible pairs of health problems between them. Then, for each single problem from u, we consider only the health problem of u' with the maximum similarity.

Definition 6 (SemS)**.** *Let u and u′ be two patients in U. The similarity based on semantic information between u and u′ is defined as*

$$SemS(u,u') = \frac{\sum_{i\epsilon Problems(u)} ps(i,u')}{|Problems(u)|} \tag{6}$$

where

$$ps(i,u') = max(\forall_{j\epsilon Problems(u')}\{simN(i,j)\}) \tag{7}$$

Instead of the maximum function used in the above process, one can employ the average function. However, according to our experiments, such an approach leads to a large number of unrelated pairs of health problems.

3.2.3. Similarity Based on Education and Health Literacy Level

Nowadays, there are a lot of sources where users can receive information about their health problems. These sources can vary in terms of how complex and how in-depth they go to showcase the problem. A user will be more attractive to sources that are inline with his/her health literacy and education level. For example, a patient with a low health literacy score will not be interested in a document that describes their health problem in great detail, but will be drawn to a document with a clear description of how to manage it. On the other hand, a patient with a high literacy score will be far more interested in the first document.

For documents regarding the same information, people have similar interests in health documents that require the same educational and health literacy level to be comprehended. As such, the similarity between two patients is calculated by the Euclidean distance between their corresponding values.

$$EducStatusS(u,u') = 1 - \frac{\sqrt{(HLit(u) - HLit(u'))^2 + (EducLvl(u) - EducLvl(u'))^2}}{\sqrt{2 * maxDif^2}} \tag{8}$$

$HLit(u)$ is a function that reports the health literacy level of user u and $Educlvl(u)$ reports his/her education level. To better combine these scores with the ratings and health problems similarity scores, we normalize them so that the function returns values in the range of $[0,1]$. The variable $maxDif$ represents the maximum difference between the two education or health literacy scores. Finally, as we want the similarity score and not the distance between the users we subtract the distance score from 1.

3.2.4. Similarity Based on Psycho-Emotional Status

Finally, anxiety and cognitive closure have an important impact on the documents preferred by people in specific periods of time, as anxiety and cognitive closure can change over time. As such, we use the Euclidean distance between the values of those two properties. As psychoemotional questionnaires are being answered periodically, we consider each time only the latest measurements on these.

$$PhychStatusS(u,u') = 1 - \frac{\sqrt{(Anxiety(u) - Anxiety(u'))^2 + (CognCl(u) - CognCl(u'))^2}}{\sqrt{2 * maxDif^2}} \tag{9}$$

$Anxiety(u)$ is a function that provides the anxiety level of user u and $CognCl(u)$ provides his/her cognitive closure status. Similarly with the similarity based on education and health literacy levels, we normalize the euclidean score and subtract it from 1 to get the similarity score.

3.2.5. Similarity between Users

Having defined all the different methods to compute similarity scores between two users, we need a way to combine all the different values into a final similarity score. We propose that not all different information perspectives are equally important to all aspect of collaborative filtering, so we assign weights on each similarity score which determines their significance.

$$S(u,u') = \alpha * RatS(u,u') + \beta * SemS(u,u') + \gamma * EducStatusS(u,u') + \delta * PhychStatusS(u,u') \quad (10)$$

where $\alpha + \beta + \gamma + \delta = 1$.

3.3. Single User Rating Model

Let P_u define the set of the most similar patients to u. Here, we refer to P_u as the *peers* of u. Formally:

Definition 7 (Peers). *Let U be a set of patients. The peers P_u of a patient $u \in U$ include the patients $u' \in U$ that are similar to u with respect to a similarity function $S(u,u')$ and a threshold δ, that is, $P_u = \{u' \in U : S(u,u') \geq \delta\}$.*

Given a patient u and his/her peers P_u, if u has no liking for a document i, the relevance of i for u is computed as

$$relevance(u,i) = \mu_u + \frac{\sum_{u' \in (P_u \cap U(i))} S(u,u')(r(u',i) - \mu_{u'})}{\sum_{u' \in (P_u \cap U(i))} |s(u,u')|} \quad (11)$$

where μ_u denotes the mean of the ratings in $I(u)$. Typically, after computing the relevance scores of the unrated documents for a user u, the documents A_u with the top-k scores are presented to u.

4. Group Recommendations

We are not only interested in recommending valuable suggestions to single patients, but to groups of patients via the caregivers who are responsible for the groups. Specifically, we focus on suggestions that are both related and fair to the group members. In Section 3.2, we discussed about the similarity functions and the relevance function was mentioned in Section 3.3. In this section, we will examine four different aggregation methods.

4.1. Group Rating Model

Typically, the related work in recommender systems targets at satisfying the interests of individual users. Recently, group recommenders that produce suggestions for groups of users (see, e.g., [29,52]) that are in the focus of the research literature. Commonly, group recommenders predict relevance scores for the unrated items for each group member, separately, and aggregate these scores to estimate the suggestions for the group. Formally, the relevance of an item for a group is defined as follows.

Definition 8 (Relevance). *Let U be a set of patients and I be a set of documents. Given a group of patients G, $G \subseteq U$, the group relevance of a document $i \in I$ for G, such that, $\forall u \in G$, $\nexists rating(u,i)$, is*

$$relevanceG(G,i) = Aggr_{u \in G}(relevance(u,i)) \quad (12)$$

With respect to the items relevance scores, the items with the top-k best scores for the group are reported to the group.

4.2. Fairness in Group Recommendations

In this work, our aim is to identify and suggest documents highly related and fair to the patients of the group. Specifically, given set of recommendations for a group to its caregiver, it is possible to have a patient u that is the least satisfied one in the group for all documents in the recommendations list, that is, all items are not relevant to u. That is, this set of documents is not fair to u. In real life, the caregiver is responsible for the needs of all group patients, and the recommender should suggest documents that are relevant and fair to the majority of the group. Inspired by work in [30], to increase the quality of the recommendations, we exploit a fairness definition that evaluates the quality of the recommendations set. Therefore, given a patient u and a set of recommendations D, we define the degree of fairness of D for u as

$$fairness(u, D) = \frac{|X|}{|D|} \tag{13}$$

where $X = A_u \cap D$. Remember, A_u are the items with the top-k relevance scores for u. Note that we only consider the intersection of the two lists as only those are going to be given to the patient. The group list is actually suggested to a caregiver, who then distributes the documents to the rest of the group according to how relevant they are to each patient. This is also why we do not take into account the ranking of each document in the group recommendation list.

To better determine the group cohesion and to understand if any member of the group is biased against, we define the *group discord* as the difference between the maximum and minimum fairness in the group.

$$groupDiscord(G, D) = max_{u \in G} fairness(u, d) - min_{u \in G} fairness(u, d) \tag{14}$$

The *group discord* takes values from 0 to 5. Ideally, we want *group discord* to take low values, as this will mean that the member of the group are treated equally. High values will indicate that at least one member is not as satisfied as the rest.

4.3. Aggregation Designs

For the aggregation method *Aggr*, we employ four different designs, each one carrying different semantics. Specifically, we divide the designs into the score-based and rank-based ones.

Score-based design predictions for documents are calculated with respect to the relevance of the documents for the group members.

In the case of the *average aggregation method*, our goal is to indulge the the majority of the group and report the average relevance for each document. Namely, relevance is computed as

$$relevanceG(G, i) = \sum_{u \in G} relevance(u, i) / |G| \tag{15}$$

In turn, a *rank-based design* aggregates the patients recommendations lists using the positions of their elements. Here, we follow the Borda count method [53], based on which each document gets 1 point for each last place in the ranking, 2 points for each next to last place, and so forth, all the way up to k points for the first place in the ranking. The document with the more points takes the first position in the list, the item with the next more points gets the second position, and so on, up to collect the best k items. The points of each document i for the group G is calculated as follows,

$$points(G, i) = \sum_{u \in G} (k - (p_u(i) - 1)), \tag{16}$$

where $p_u(i)$ defines the position of item i in A_u.

The *Fair* method [11] belongs as well to the *rank-based* methods. Fair considers pairs of patients in the group to make predictions. Specifically, a document i belongs to the top-k suggestions for a group G, if for a pair of patients $u_1, u_2 \in G$, $i \in A_{u_1} \bigcap A_{u_2}$, and i is the document with the maximum rank in A_{u_2}.

To produce recommendations, Fair incrementally creates an initially empty set D by choosing for each pair of patients u_x and u_y, the document in A_{u_x} with the maximum relevance score for u_y (Algorithm 1). If k (i.e., documents to be reported to the group) is greater than the documents, we are able to find recommendations using the method above: we add documents to D by iterating the A_u lists of the group members and adding each time the document with the maximum rank that does not appear in D.

Algorithm 1: Fair Group Recommendations Algorithm

```
Data: A group of users $G = \{u_1, \ldots, u_n\}$, the sets of recommendations $A_{u_x}$, $\forall u_x \in G$.
Result: The $z$ documents in the recommendations list $D$ for $G$.
1  $D \leftarrow \emptyset$
2  while $|D| < z$ do
3      for $x = 0; x < n; x{+}{+}$ do
4          for $y = 0; y < n; y{+}{+}$ do
5              if $x \neq y$ then
6                  Locate the document $i \in A_{u_y}$ with the max $relevance(u_x, i)$
7                  $D = D \cup i$
8              end
9          end
10     end
11 end
12 return $D$
```

In addition, we propose a new aggregation method, called *AccScores* method, which is inspired by the Borda method, but instead of accumulating the points of each item, we accumulate the scores of the items. We add the scores as they appear in the A_u of all the group members in a set called *accDoc*. The first item we select to include in the group recommendation list is the one with the highest score in *accDoc*. After each selection, we update a helper structure *accUser* that consists of the users and their accumulating preference scores. For each user, we accumulate the scores of the items that were selected as they appear in the individual preference list A_u. If there is a user u that has a lower score than the rest, in the next selection, we will choose an item that exists in the A_u and at the same time has the highest possible score in the *accDoc*. If many users have the same lowest score, we select the user that has been chosen the least amount of times. This process is shown in Algorithm 2.

In Lines 1–10, we populate the sets *accDoc* and *accUser*. If all the users have the same accumulated score (Line 12), then we select the item with the highest score in *accDoc* (Line 13). Otherwise, we find the user with the lowest score (Line 15), and then we locate the item that appears both in the user's preference list and has the highest possible score in *accDoc* (Line 16). Then, we add to the structure *accUser* the score of the selected item for each member (Lines 18-20). Finally, we include the item in the group recommendation list D.

Algorithm 2: AccScores Group Recommendations Algorithm

```
Data: A group of users $G = \{u_1, \dots, u_n\}$, the sets of recommendations $A_{u_x}$ ,$\forall u_x \in G$.
Result: The $z$ items in the recommendations list $D$ for group $G$.
for $u \in G$ do
    for $i \in A_u$ do
        if $i \in accDoc$ then
            $accDoc[i] = accDoc[i] + relevance(u, i)$ ;
        else
            $accDoc[i] = relevance(u, i)$ ;
        end
    end
    $accUser[u] = 0$;
end
while $|D| < z$ do
    if $allEqual(accUser)$ then
        $id = addTop(accDoc)$;
    else
        $p = findMinUser(accUser)$;
        $id = addMaxDoc(A_p, accDoc)$;
    end
    for $u \in G$ do
        $accUser[u] = accUser[u] + relevance(u, id)$;
    end
    $D = D \cup id$;
end
return $D$;
```

5. Dataset

Nowadays, it is quite common for patients to search for information related to their health problems, as well as to rate the related documents that appear on the Web. However, the profiles of such patients are not accessible and linked to those documents. For several reasons, including ethical and legal constraints, the collection and use of such a data is prohibited.

To experiment with such a dataset, we initially exploited 10,000 chimeric patient profiles [54]. These profiles contain characteristics similar to the ones existing in a real medical database. For example, we consider the patients' admission details, demographics, socioeconomic details, labs, and medications. Additionally, we use the ICD10 ontology for describing the health problems for each patient, making this dataset ideal for our semantic similarity approach.

Then, by exploiting these profiles, we create a synthetic dataset that includes a document corpus and user ratings. Specifically:

- Document Corpus
  - *Create document corpus.* Initially, we generated *numDocs* documents for each node in the second level of the ontology tree that represents the ICD10 ontology. For each such document, we selected randomly *numKeyWords* words from the nodes descriptions in each subsequent subtree.
  - *Assignment of Education and Health Literacy Levels.* We divide the documents based on five percentage scores $eduHLit_1 \dots eduHLit_5$ that correspond to the five different education levels. We assign to the documents in each subgroup their corresponding education level. We propose that a document cannot have a vastly different education and health literacy

score. A document that has high education level is improbable to be for users with low literacy score and, similarly, a document with high health literacy is not probable to have a low education level. Therefore, with equal probability, we assign to each document a health literacy score that is the same, one highest or one lowest level than that of its education level.

- Rating Dataset
  - *Divide the patients into groups.* We assume that all patients have assigned $numRatings$ ratings to documents. For doing so, we distinguish the patients between $occasional$, $regular$, and $dedicated$. The users in each group gave *few*, *average* and *a lot* of ratings, respectively.
  - *Assignment of Education and Health Literacy Levels.* The procedure to assign education and health literacy levels to the patients is the same as the one to assign them to the documents.
  - *Assignment of Anxiety and Cognitive Closure.* Anxiety and cognitive closure scores are regularly measured for each patient since these tend to change rapidly. This is why in our methods we only take into account the most recent ones. Therefore, in our dataset, we generate one anxiety and cognitive closure score for each patient. We follow a similar method as the one for education and health literacy levels and divide the patients based on five percentage scores $AnxCognCl_1 \ldots AncCognCl_5$. However, now anxiety will be the score that will define cognitive closure. The more anxious a person is about their health problems the more he/she needs to understand them.
  - *Simulate a power law rating distribution.* When ranking documents with respect to real users preferences, the documents typically follow the power law distribution. To show this, we randomly chose $popularDocs$ documents and consider them as the most popular.
  - *Generate documents to rate.* For each patient, we distinguished the ratings that he/she will give between $healthRelevant$ and $nonRelevant$. Given the assumption that patients are interested in both documents related to their health problems, as well as to other documents, we assigned ratings to both such groups of documents.
  - *Generate ratings.* Last, for each item generated above, we randomly assigned *a rating* from 1 to 5.

The parameters that were used to generate the datasets needed for our experiments are shown is Tables 3 and 4, which contain the parameters for the document corpus and rating dataset, respectively. The education percentages $eduHLit_1 \ldots eduHLit_5$ are only showcased in Table 4, but the same values were used for the generation of document corpus.

Table 3. Input parameters for generating the document corpus.

Parameter Name	Explanation	Value
numDocs	# of documents generated for each category of health problems.	200
numKeyWords	# of keywords appended to documents.	10
popularDocs	The # of the most popular documents in each category, for simulating a power law distribution.	70

Table 4. Input parameters for generating the ratings dataset.

Partitions	Parameter Name	Explanation	Value
Group Partition	Group *occasional*	# of ratings given by patients in this group is 20 to 100	50% of all patients
	Group *regular*	# of ratings given by patients in this group is 100 to 250	30% of all patients
	Group *dedicated*	# of ratings given by patients in this group is 250 to 500	20% of all patients
Education Levels	$EduHLit_1$	Patients with Education Level 1	5% of all patients
	$EduHLit_2$	Patients with Education Level 2	10% of all patients
	$EduHLit_3$	Patients with Education Level 3	40% of all patients
	$EduHLit_4$	Patients with Education Level 4	30% of all patients
	$EduHLit_5$	Patients with Education Level 5	15% of all patients
Anxiety Scores	$AnxCognCl_1$	Patients with Anxiety Score 1	30% of all patients
	$AnxCognCl_2$	Patients with Anxiety Score 2	40% of all patients
	$AnxCognCl_3$	Patients with Anxiety Score 3	15% of all patients
	$AnxCognCl_4$	Patients with Anxiety Score 4	10% of all patients
	$AnxCognCl_5$	Patients with Anxiety Score 5	5% of all patients
Scores Partition	One	# of ratings that have as score 1	20% of all ratings
	Two	# of ratings that have as score 2	10% of all ratings
	Three	# of ratings that have as score 3	30% of all ratings
	Four	# of ratings that have as score 4	20% of all ratings
	Five	# of ratings that have as score 5	20% of all ratings
Ratings Partition	healthRelevant	# of relevant to some health problems documents each user will rate	40% of ratings from each user
	nonRelevant	# of non relevant to any health problems documents each user will rate.	60% of ratings from each user

6. Evaluation

In this section, we present the metrics we used for the experimental evaluation of the similarities functions and aggregation methods as well as the results of these evaluations.

6.1. Evaluation Measures

To evaluate the similarity functions, we used the normalized Discounted Cumulative Gain [55]. The $nDCG$ values for all users' recommendation lists can be averaged to get a measure of the average performance of a recommendation system. The $nDCG$ can be calculated as follows,

$$nDCG_u = \frac{DCG_u}{IDCG_u} \tag{17}$$

where

$$DCG_u = \sum_{i=1}^{k} \frac{2^{relevance(u,i)} - 1}{log_2(i+1)} \tag{18}$$

and

$$IDCG_u = \sum_{i=1}^{k} \frac{2^{r(u,i)} - 1}{log_2(i+1)} \tag{19}$$

The DCG_u part of the equation calculates the relevance of the items that appear in the recommendation list of a user and the $IDCG_u$ calculates the relevance of the items in an ideal scenario. Note that in an ideal recommendation list, DCG_u equals to $IDCG_u$ producing an nDCG of 1.0. Then, nDCG scores are relative values on the interval 0.0 to 1.0.

To evaluate the top-k results of each aggregation method, we counted the average of the distance between the top-k recommendation list produced for the group and the list produced for each user separately. For computing the distance, we used the Kendall tau distance that numerates the number of pairwise disagreements between two ranking lists [56].

$$\begin{aligned} K(t_1, t_2) = &|\{(i,j) : i < j, (t_1(i) < t_1(j) \wedge t_2(i) > t_2(j)) \\ &\vee (t_1(i) > t_i(j) \wedge t_2(i) < t_2(j))\}| \end{aligned} \tag{20}$$

where $t_1(i)$ and $t_2(i)$ are the rankings of the element i in t_1 and t_2, respectively.

6.2. Evaluation Results

6.2.1. Evaluation of Similarity Functions

To evaluate the proposed similarity functions, we used the recommendations produced for single users. We used 50 users, for which we hidden 20% percent of their ratings. We then applied the recommendation algorithm using different values for the variables α, β, γ, and δ and predicted a score for them. We used the hidden items as the ground truth for the calculation of the $IDCG$. Finally, we averaged these scores. The results are shown in Figure 2. In our experiments, for computing the semantic similarity function *SemS*, we used the value of 0.1 for constant w that is needed in Definition 1. As a reminder, α is the weight that corresponds to the rating similarity *RatS*, β to health problems similarity *SemS*, γ to education/health literacy similarity *EducStatusS*, and δ to anxiety/cognitive closure similarity *PhychStatusS*.

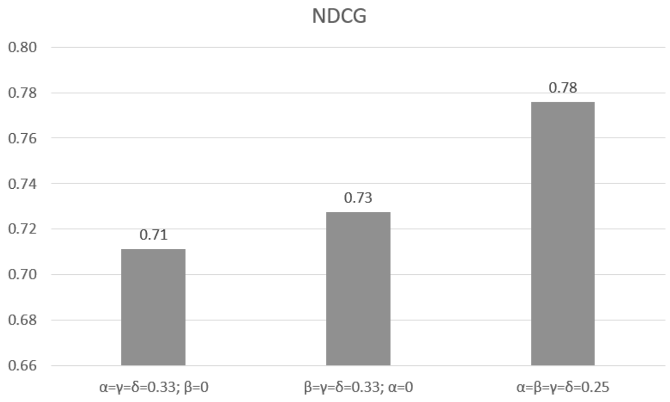


Figure 2. nDCG values for different values of α, β, γ, and δ.

In our previous work [11], we showed that *SemS* outperforms *RatS*. We now want to focus on what effect the *EducStatusS* and *PhychStatusS* has on them. When we introduce the two new similarities to the old ones, we can still observe that the *SemS* gives better results. However, if we combine all the similarities we get the best $nDCG$ values. The *SemS* and *RatS* similarities can compensate for the faults of each other. *SemS* can find patients with similar health problems which means that they have an interest in the same documents. *RatS* can find all the other patients that are not necessarily related with similar health problems, but have a similar interest in documents. The results improve further when we *EducStatusS* and *PhychStatusS*. They further refine the selection of the peers, so the recommendations are more accurate.

We did not make any evaluations for γ and δ on their own as the *EducStatusS* and *PhychStatusS* similarities offer more auxiliary and not defining information for the patients. They need another similarity to augment their knowledge in order to function as intended.

6.2.2. Evaluation of Aggregation Methods

To evaluate the effectiveness of each aggregation method, in regards to the construction of the final group recommendation list, we use the Kendall tau distance. In more detail, we calculate the distance between the *top-k* list of each group member, and the recommendation list for the group. Additionally, we have normalized all distance scores to the range of [0,1]. Intuitively, a low distance score between the two lists, meaning that the recommender system suggests close to optimal items for that user. In this way, we can estimate the difference between the lists, and consequentially how many of the most highly recommended documents for each user, have been included in the group recommendations. These experiments help us identify whether each aggregation method makes adequate use of the individual top-k list of the members of the group.

To produce these results, we selected randomly 40 different groups with approximately the same *group similarity*. Group similarity is defined as the summation of the similarity of all pairs of users in the group, averaged over the number of the pairs. As our working study case is for a care provider responsible for a group of patients, it makes sense for these patients to be similar. Furthermore, we propose that during the formation of the groups the most important factor is the health similarity followed by their education/health literacy similarity. Their anxiety levels (which are ephemeral) or their ratings (personal preferences) are not of equal importance when we want to group people that will be cared for by just one person. We assign the following weights to Equation (10): $\alpha = 0.25$, $\beta = 0.4$, $\gamma = 0.3$, $\delta = 0.05$.

After generating the group recommendation list, we compute for each group member the distance between the individual top-k list and the group recommendation list. The distance score for one group for each aggregation method is the average score of the sum of these distances over the size of the group. After following the same process for all 40 groups, the overall score for each aggregation method is the mean of the previously calculated scores, over the number of groups. We set k to be 20, meaning the group recommendation list consists of 20 items. Figures 3 gives the results for groups with similarity 0.6. This figure will give a general overview of the effectiveness of each aggregation design.

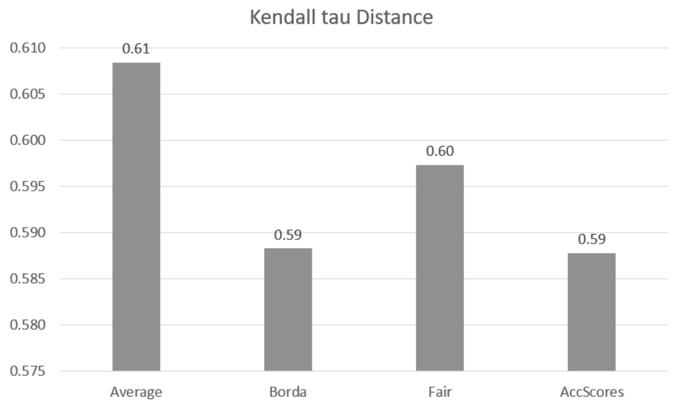


Figure 3. The Kendal tau Distances for groups with group similarity 0.6.

We can see that both the Borda and AccScores aggregation methods perform the best, followed by the Fair method. Average has the worst results. However, the differences between the aggregation methods are minuscule. Due to our case study, all the group members are similar to each other to a degree. When trying to aggregate their top-k recommendations, regardless of the aggregation design, the most relevant items to the group are suggested. Additionally, as we calculate the average distance score for each group, it is expected to have higher values of distance scores for each method. Although the methods identify many of the users relevant documents, the high distance scores mostly correspond to the difference in the positions of the items between the two lists. What we are more interested in is the individual satisfaction of each group member. Remember this group does not ask for recommendations before proceeding to make a decision. We give a list of recommended items to a carer who proceeds to distribute them to a group of patients that he/she is responsible for.

To better understand the individual impact of the recommendation list to each group member, we have calculated the *group discord*. Ideally, we want all the members to be treated equally, meaning that the group recommendation list should be fair to all group members. This is especially important in the health domain, where information about people's health should be as accurate as possible. In that regard, the system should not return a list that is biased against one member. Therefore, the lower *group discord* values an aggregation method generates, the better it is for our purposes.

Figure 4 shows the *group discord* values for the same 40 groups we used in the previous experiment. Even though the previous experiment did not show any huge difference in the behavior of the aggregation methods, when we compare their fairness aspect, there is a distinct disparity between them. This higher variance in the group discord scores of the different aggregation methods compared to the ones from the distance measure is attributed to the different natures of the two measures. Kendall tau distance not only takes into account the existence of an item in the two lists, but also their position. On the other hand, our fairness method (Equation (13)) only considers if two items are present in both lists.

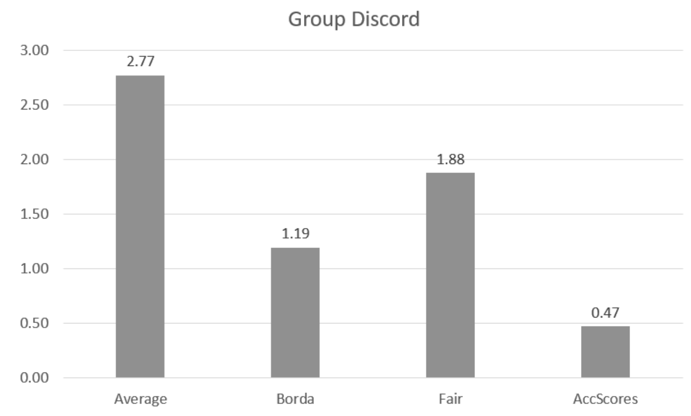


Figure 4. Group Discord for groups with group similarity 0.6.

The AccScores method manages to identify a set of items that are almost equally fair to all members (*group discord* is lower than 0.5), whereas the Average method has the worst results with values above 2.5. Borda and Fair methods offer median results, with Borda being slightly better than Fair. This experiment makes more apparent the advantage of the AccScores method over all the rest. Even though in the previous experiment it had the same scores as the Borda method, AccScores manages to be fairer to the members of the group. For our case study, being fair to all members is a top priority for any system.

7. Conclusions

In this work, we focus on multidimensional group recommendations in the health domain, using collaborative filtering. For identifying similarity among patients, we go beyond ratings to also consider the medical problems, the education, the health literacy, and the psycho-emotional statues of the patients, all available in their personal profile. Based on those dimensions, we introduce a new aggregation method accumulating preference scores and we experimentally show that it manages to identify set of items that are almost equally fair to all members of the group.

The semantic similarity measure proposed assumes that the health information of a patient is captured using standard terminologies. Although this is a common practice nowadays, there is still a lot of textual information that are not always mapped to standard terminologies. Nevertheless, today there exist many tools that annotate effectively textual descriptions to terminological terms. For example, the Bioportal Annotator (https://bioportal.bioontology.org/annotator) exposes programmatically an API for annotating textual information with multiple terminologies. An extension of our work could use this API to annotate textual descriptions as well. The same assumption holds for the interesting documents recommended to the patients. Additionally, as future work, we intend to explore whether introducing additional patient characteristics (e.g., gender, stress, and medications) to our recommendation model can further improve the quality of the recommendations.

Author Contributions: Conceptualization, M.S., H.K., and K.S.; Data curation, M.S.; Formal analysis, M.S., H.K., and K.S.; Investigation, M.S., H.K., and K.S.; Methodology, M.S.; Software, M.S.; Supervision, H.K. and K.S.; Validation, M.S.; Writing—original draft, M.S.; Writing—review & editing, M.S., H.K., and K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berg, G.M.; Hervey, A.M.; Atterbury, D.; Cook, R.; Mosley, M.; Grundmeyer, R.; Acuna, D. Evaluating the quality of online information about concussions. *J. Am. Acad. PAS* **2014**, *27*, 1547–1896. [CrossRef] [PubMed]
2. McMullan, M. Patients using the Internet to obtain health information: How this affects the patient-health professional relationship. *Patient Educ. Couns.* **2006**, *63*, 24–28. [CrossRef] [PubMed]
3. Wiesner, M.; Pfeifer, D. Adapting recommender systems to the requirements of personal health record systems. In Proceedings of the ACM International Health Informatics Symposium, IHI 2010, Arlington, VA, USA, 11–12 November 2010; pp. 410–414.
4. Brandon, I.; Daniel, K.; Patrice, C.; Nicholas, T. Testing the Efficacy of OurSpace, a Brief, Group Dynamics-Based Physical Activity Intervention: A Randomized Controlled Trial. *J. Med. Internet Res.* **2016**, *18*, e87.
5. Cheung, D.Y.T.; Chan, H.C.H.; Lai, J.C.K.; Chan, V.W.F.; Wang, P.M.; Li, W.H.C.; Chan, C.S.S.; Lam, T.H. Using WhatsApp and Facebook Online Social Groups for Smoking Relapse Prevention for Recent Quitters: A Pilot Pragmatic Cluster Randomized Controlled Trial. *J. Med. Internet Res.* **2015**, *17*, e238. [CrossRef] [PubMed]
6. Batenburg, A.; Das, E. Emotional Approach Coping and the Effects of Online Peer-Led Support Group Participation Among Patients With Breast Cancer: A Longitudinal Study. *J. Med. Internet Res.* **2014**, *16*, e256. [CrossRef]
7. Serbos, D.; Qi, S.; Mamoulis, N.; Pitoura, E.; Tsaparas, P. Fairness in Package-to-Group Recommendations. In Proceedings of the 26th International Conference on World Wide Web (WWW), Perth, Australia, 3–7 April 2017.
8. Machado, L.; Stefanidis, K. Fair Team Recommendations for Multidisciplinary Projects. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki, Greece, 14–17 October 2019.
9. Stratigi, M.; Kondylakis, H.; Stefanidis, K. Fairness in Group Recommendations in the Health Domain. In Proceedings of the 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, 19–22 April 2017; pp. 1481–1488.
10. Stratigi, M.; Kondylakis, H.; Stefanidis, K. The FairGRecs Dataset: A Dataset for Producing Health-related Recommendations. In Proceedings of the First International Workshop on Semantic Web Technologies for Health Data Management, SWH@ISWC 2018, Monterey, CA, USA, 9 October 2018.
11. Stratigi, M.; Kondylakis, H.; Stefanidis, K. FairGRecs: Fair Group Recommendations by Exploiting Personal Health Information. In Proceedings of the Database and Expert Systems Applications—29th International Conference, DEXA 2018, Regensburg, Germany, 3–6 September 2018; pp. 147–155.
12. Kondylakis, H.; Bucur, A.; Crico, C.; Dong, F.; Graf, N.; Hoffman, S.; Koumakis, L.; Manenti, A.; Marias, K.; Mazzocco, K.; et al. Patient empowerment for cancer patients through a novel ICT infrastructure. *J. Biomed. Inform.* **2020**, *101*, 103342. [CrossRef]
13. Kondylakis, H.; Koumakis, L.; Katehakis, D.G.; Kouroubali, A.; Marias, K.; Tsiknakis, M.; Simos, P.G.; Karademas, E. Developing a Data Infrastructure for Enabling Breast Cancer Women to BOUNCE Back. In Proceedings of the 32nd IEEE International Symposium on Computer-Based Medical Systems, CBMS 2019, Cordoba, Spain, 5–7 June 2019; pp. 652–657.
14. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl.-Based Syst.* **2013**, *46*, 109–132. [CrossRef]

15. Mooney, R.J.; Roy, L. Content-based book recommending using learning for text categorization. In Proceedings of the Fifth ACM Conference on Digital Libraries, San Antonio, TX, USA, 2–7 June 2000; pp. 195–204. [CrossRef]
16. Konstan, J.A.; Miller, B.N.; Maltz, D.; Herlocker, J.L.; Gordon, L.R.; Riedl, J. GroupLens: Applying Collaborative Filtering to Usenet News. *Commun. ACM* **1997**, *40*, 77–87. [CrossRef]
17. Lu, J.; Wu, D.; Mao, M.; Wang, W.; Zhang, G. Recommender system application developments: A survey. *Decis. Support Syst.* **2015**, *74*, 12–32. [CrossRef]
18. Yin, Z.; Gupta, M.; Weninger, T.; Han, J. LINKREC: A unified framework for link recommendation with user attributes and graph structure. In Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, NC, USA, 26–30 April 2010.
19. Eirinaki, M.; Abraham, S.; Polyzotis, N.; Shaikh, N. QueRIE: Collaborative Database Exploration. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1778–1790. [CrossRef]
20. Koskela, M.; Simola, I.; Stefanidis, K. Open Source Software Recommendations Using Github. In Proceedings of the International Conference on Theory and Practice of Digital Libraries, Porto, Portugal, 10–13 September 2018.
21. Ge, X.; Chrysanthis, P.K.; Pelechrinis, K. MPG: Not So Random Exploration of a City. In Proceedings of the 17th IEEE International Conference on Mobile Data Management (MDM), Porto, Portugal, 13–16 June 2016.
22. Stratigi, M.; Nummenmaa, J.; Pitoura, E.; Stefanidis, K. Fair Sequential Group Recommendations. In Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing, SAC 2020, Brno, Czech Republic, 30 March–3 April 2020.
23. Borges, R.; Stefanidis, K. Enhancing Long Term Fairness in Recommendations with Variational Autoencoders. In Proceedings of the 11th International Conference on Management of Digital EcoSystems, MEDES 2019, Limassol, Cyprus, 12–14 November 2019.
24. Stefanidis, K.; Kondylakis, H.; Troullinou, G. On Recommending Evolution Measures: A Human-Aware Approach. In Proceedings of the 33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, 19–22 April 2017; pp. 1579–1581.
25. Troullinou, G.; Kondylakis, H.; Stefanidis, K.; Plexousakis, D. Exploring RDFS KBs Using Summaries. In Proceedings of the The Semantic Web—ISWC 2018—17th International Semantic Web Conference, Monterey, CA, USA, 8–12 October 2018; pp. 268–284.
26. Gartrell, M.; Xing, X.; Lv, Q.; Beach, A.; Han, R.; Mishra, S.; Seada, K. Enhancing group recommendation by incorporating social relationship interactions. In Proceedings of the 2010 International ACM SIGGROUP Conference on Supporting Group Work, GROUP 2010, Sanibel Island, FL, USA, 6–10 November 2010; pp. 97–106.
27. Castro, J.; Lu, J.; Zhang, G.; Dong, Y.; Martínez-López, L. Opinion Dynamics-Based Group Recommender Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, *48*, 2394–2406. [CrossRef]
28. Amer-Yahia, S.; Roy, S.B.; Chawla, A.; Das, G.; Yu, C. Group Recommendation: Semantics and Efficiency. *PVLDB* **2009**, *2*, 754–765. [CrossRef]
29. Ntoutsi, E.; Stefanidis, K.; Nørvåg, K.; Kriegel, H. Fast Group Recommendations by Applying User Clustering. In Proceedings of the Conceptual Modeling—31st International Conference ER 2012, Florence, Italy, 15–18 October 2012; pp. 126–140.
30. Qi, S.; Mamoulis, N.; Pitoura, E.; Tsaparas, P. Recommending Packages to Groups. In Proceedings of the IEEE 16th International Conference on Data Mining, ICDM 2016, Barcelona, Spain, 12–15 December 2016; pp. 449–458.
31. Pfeifer, D. Health Recommender Systems: Concepts, Requirements, Technical Basics and Challenges. *Int. J. Environ. Res. Public Health* **2014**, *11*, 2580–2607.
32. Schäfer, H.; Hors-Fraile, S.; Karumur, R.P.; Valdez, A.C.; Said, A.; Torkamaan, H.; Ulmer, T.; Trattner, C. Towards Health (Aware) Recommender Systems. In Proceedings of the 2017 International Conference on Digital Health, London, UK, 2–5 July 2017; pp. 157–161.
33. Hors-Fraile, S.; Romero, O.R.; Schneider, F.; Fernández-Luque, L.; Luna-Perejón, F.; Balcells, A.C.; de Vries, H. Analyzing recommender systems for health promotion using a multidisciplinary taxonomy: A scoping review. *Int. J. Med Inform.* **2018**, *114*, 143–155. [CrossRef]
34. Agapito, G.; Calabrese, B.; Guzzi, P.H.; Cannataro, M.; Simeoni, M.; Care, I.; Lamprinoudi, T.; Fuiano, G.; Pujia, A. DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion.

In Proceedings of the 2th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2016, New York, NY, USA, 17–19 October 2016; pp. 1–8. [CrossRef]

35. Espín, V.; Hurtado, M.V.; Noguera, M. Nutrition for Elder Care: A nutritional semantic recommender system for the elderly. *Expert Syst.* **2016**, *33*, 201–210. [CrossRef]
36. Hors-Fraile, S.; Núñez-Benjumea, F.J.; Hernández, L.C.; Ruiz, F.O.; Fernández-Luque, L. Design of two combined health recommender systems for tailoring messages in a smoking cessation app. *arXiv* **2016**, arXiv:1608.07192.
37. Smileska, C.; Koceska, N.; Koceski, S.; Trajkovik, V. Development and Evaluation of Methodology for Personal Recommendations Applicable in Connected Health. In *Enhanced Living Environments—Algorithms, Architectures, Platforms, and Systems*; Lecture Notes in Computer Science; Ganchev, I., Garcia, N.M., Dobre, C., Mavromoustakis, C.X., Goleva, R., Eds.; Springer: New York, NY, USA, 2019; Volume 11369, pp. 80–95.
38. Afolabi, A.O.; Toivanen, P.J. Integration of Recommendation Systems Into Connected Health for Effective Management of Chronic Diseases. *IEEE Access* **2019**, *7*, 49201–49211. [CrossRef]
39. Hu, H.; Elkus, A.; Kerschberg, L. A Personal Health Recommender System incorporating personal health records, modular ontologies, and crowd-sourced data. In Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016, San Francisco, CA, USA, 18–21 August 2016; pp. 1027–1033.
40. Torrent-Fontbona, F.; López, B. Personalized Adaptive CBR Bolus Recommender System for Type 1 Diabetes. *IEEE J. Biomed. Health Inform.* **2019**, *23*, 387–394. [CrossRef]
41. Kim, J.; Lee, D.; Chung, K. Item recommendation based on context-aware model for personalized u-healthcare service. *Multimed. Tools Appl.* **2014**, *71*, 855–872. [CrossRef]
42. Wang, S.; Chen, Y.; Kuo, A.M.; Chen, H.; Shiu, Y. Design and evaluation of a cloud-based Mobile Health Information Recommendation system on wireless sensor networks. *Comput. Electr. Eng.* **2016**, *49*, 221–235. [CrossRef]
43. Kondylakis, H.; Koumakis, L.; Kazantzaki, E.; Chatzimina, M.; Psaraki, M.; Marias, K.; Tsiknakis, M. Patient Empowerment through Personal Medical Recommendations. In Proceedings of the MEDINFO 2015: EHealth-enabled Health—Proceedings of the 15th World Congress on Health and Biomedical Informatics, São Paulo, Brazil, 19–23 August 2015; p. 1117.
44. Kondylakis, H.; Koumakis, L.; Psaraki, M.; Troullinou, G.; Chatzimina, M.; Kazantzaki, E.; Marias, K.; Tsiknakis, M. Semantically-enabled Personal Medical Information Recommender. In Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, 11 October 2015.
45. Nores, M.L.; Blanco-Fernández, Y.; Pazos-Arias, J.J.; Gil-Solla, A. Property-based collaborative filtering for health-aware recommender systems. *Expert Syst. Appl.* **2012**, *39*, 7451–7457. [CrossRef]
46. Qin, L.; Xu, X.; Li, J. A Real-Time Professional Content Recommendation System for Healthcare Providers' Knowledge Acquisition. In Proceedings of the Big Data—BigData 2018—7th International Congress, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, 25–30 June 2018; pp. 367–371.
47. Bermingham, A.; Caprani, N.; Collins, R.; Gurrin, C.; Irving, K.; O'Rourke, J.; Smeaton, A.F.; Yang, Y. Recommending Video Content for Use in Group-Based Reminiscence Therapy. In *Health Monitoring and Personalized Feedback Using Multimedia Data*; Springer: New York, NY, USA, 2015; pp. 215–244.
48. Kondylakis, H.; Kazantzaki, E.; Koumakis, L.; Genitsaridi, I.; Marias, K.; Gorini, A.; Mazzocco, K.; Pravettoni, G.; Burke, D.; McVie, G. et al. Development of interactive empowerment services in support of personalised medicine. *eCancer* **2014**, *8*, 400. [CrossRef] [PubMed]
49. Berkman, N.D.; Davis, T.C.; McCormack, L. Health Literacy: What Is It? *J. Health Commun.* **2010**, *15*, 9–19. [CrossRef] [PubMed]
50. Berkman, N.D.; DeWalt, D.A.; Pignone, M.; Sheridan, S.L.; Lohr, K.N.; Lux, L.; Sutton, S.F.; Swinson, T.; Arthur, J. Literacy and health outcomes. *J. Gen. Intern. Med.* **2004**, *19*, 1228–1239.
51. Lillie, S.E.; Fu, S.S.; Fabbrini, A.E.; Rice, K.L.; Clothier, B.A.; Doro, E.; Melzer, A.C.; Partin, M.R. Does need for cognitive closure explain individual differences in lung cancer screening? A brief report. *J. Health Psychol.* **2018**, 1359105317750253. [CrossRef] [PubMed]
52. Roy, S.B.; Amer-Yahia, S.; Chawla, A.; Das, G.; Yu, C. Space efficiency in group recommendation. *VLDB J.* **2010**, *19*, 877–900. [CrossRef]
53. Emerson, P. The original Borda count and partial voting. *Soc. Choice Welf.* **2013**, *40*, 353–358. [CrossRef]

54. Kartoun, U. A Methodology to Generate Virtual Patient Repositories. *arXiv* **2016**, arXiv:1608.00570.
55. Järvelin, K.; Kekäläinen, J. IR Evaluation Methods for Retrieving Highly Relevant Documents. *SIGIR Forum* **2017**, *51*, 243–250. [CrossRef]
56. Kendall, M. *Rank Correlation Methods*; Griffin: London, UK, 1948.