FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

**MASTER THESIS**

Bc. Ladislav Maleček

# Fairness in group recommender systems

Department of Software Engineering

Supervisor of the master thesis: Mgr. Ladislav Peška, Ph.D.
Study programme: Computer Science
Study branch: Artificial Intelligence

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ............. date .............   ....................................

Author's signature

Dedication.

Title: Fairness in group recommender systems

Author: Bc. Ladislav Maleček

Department: Department of Software Engineering

Supervisor: Mgr. Ladislav Peška, Ph.D., Department of Software Engineering

Abstract: Abstract.

Keywords: key words

# Contents

# 1. Introduction

Most of us interact with many recommender systems daily. Even if seemingly indirectly. The proliferation of this technology is astounding. Almost every interaction with today's web is in some way personalized. From the search results, shopping, listening to music, reading news, to browsing social media, and many more. It has become quite unavoidable.

We can view recommender systems from a very simple perspective - they are algorithms that recommend items to users. Where items and users can be many different things, items, for example, being movies, news articles, a more complex object, or even entire systems. And users being, real people or other entities that exhibit some sort of preference on which the algorithm can decide.

One of the variants of recommender systems is those where the recommendation result is shared among more users based on their shared (aggregated) preferences. This is a subset called group recommender systems. They are not used as widely as the non-group variants due to the nature of the usage of most of the aforementioned technologies. We mostly use the web, listen to music and read the news as individuals. At least from the perspective of those systems. But for some of the domains, there are valid use cases. We often listen to music and watch movies in groups. Select a restaurant and other public services not just for us. And that's where the group recommenders come in handy.

tady je trochu moc velky skok - nejdriv asi neco o tom jak group RS funguji. Spis nez evaluation zminit primo fairness, nebo obecneji co ma byt cilem skupinoveho doporuceni

But how to approach the evaluation? It starts to become harder than just simply rating the results based on single feedback, now we have multiple users with possibly very different personal experiences. We want to be fair towards all individuals in the group. But the fairness property can be tricky to describe and evaluate due to the subjective nature of preference perception and distribution among the group members.

Classical recommendation systems have been studied for quite a long time, but the group variant and more soft-level (meaning evaluation on other than classical parameters) thinking about them is quite recent. With the rise of social dilemmas around recommender systems appears the fairness-ensuring topic more and more in many different shapes and sizes. And with that, there is growing popularity towards recommender systems that are trained (and therefore evaluated) with these novel requirements in mind.

## 1.1 Problem statement

The current research on the topic of group recommender systems is lacking. There are no standardized data sets that would offer evaluation of the research without using various methods of data augmentation and artificial data creation. And the definition of fairness is not unified. It can mean many different things and be evaluated with many different methods.

These two aforementioned problems go hand in hand with the very subjective nature of user preference.

Mozna merge obou kapitol dohromady?

## 1.2 Research objective

We would like to study how fairness can be defined in the context of the recommender systems, how it can be measured and eventually used to improve recommendations in the group setting. And explore different variants of fairness such as long-term fairness and different distribution of fairness among group members.

The primary goal of this thesis is to research and design a novel group recommender system algorithm that would keep fairness as its primary optimization objective. If we could adapt fairness preserving methods such as voting systems from other fields to group recommendation problem. And evaluating the new algorithm with already existing approaches in the domain of group recommender systems.

Additionally, we would like to research and contribute to data sets that could be used for the group setting. Expanding single user data sets with data augmentation that would generate synthetic groups' information and creating a web application in a movie domain that would serve as a platform for online evaluation of group recommender algorithms and provided us with real-user group recommendation data.

## 1.3 Thesis structure

We start with an introduction to recommender systems and specifically to group recommender systems in the chapter: Recommender systems. Then we will continue with the definitions and evaluation methods for fairness in chapter Fairness. Next, we will introduce few algorithms that are used in the group recommender field in chapter Related work. TODO: check out other works and decide what should be here. This can be nice from the reading perspective, but is it really necessary?

# 2. Recommender systems

In this chapter, we will briefly introduce in general what recommender systems are (hereinafter referred to as RS) and then continue with a description of the group variant of recommender systems and introduce common approaches and methods they employ.

## 2.1 Recommender systems

Broadly speaking, recommender systems are algorithms that are trying to suggest items to their users, or from another perspective, they aim to predict how would a user rate (like) an unseen item. They are used in a variety of settings, from e-commerce, media consumption, social networks, expert systems, search engines, and many others.

At their core as stated in [1] they are essentially an information filtering systems that aim to deal with selecting a subset from all the items by some filtration criteria, in this case the criteria is the user preference. They become necessary when it comes to suppressing the explosive growth in information on the web and function as a defence system against over loading the user with the vast amount of data that is present in almost every system today.

They can be views as decision support systems that guide users in finding and identifying items based on their idea about the desired state, in this situation the desired state is to find an item that they would like [2].

RS can provide both, by filtering based on user preference and providing alternatives by utilizing similarity. In a way, finding a suitable item can be viewed as a collaboration of the user and the recommender system, in varying degrees of freedom. From passively accepting the RS recommendations to actively interacting by giving feedback and stating the preferences.

### High-level examples

Recommender systems are used in multiple ways, we now present a few high-level examples of where and how they can be used.

- **Personalized merchandising**, where the system offers items that other users bought together with the viewed item, items that user could like based on previous orders or viewed items, either related or alternative choices.
- **Personalized content**, for content consumption services such as video and audio libraries. User is offered personalized content based on their preference profile, such as movies or videos that are similar to others they viewed, globally popular for a regional subset of the user-base and so on.
- **Personalized news feed and social media feed.** We want to offer user interesting content that would keep them engaged with the service. In the recent years there is push towards more social responsible RS design in this context due to the overwhelming power the social media have. It is important to deal with problems such as polarization [3], fairness and disagreement.

- **Expert systems** that help doctors, operators and other people to make an informed decisions based on data. They can help to deal with data overload and filter relevant items and choices. As well as explore the item space when searching for solution with only weakly defined requirements.
- **Search experience**, that takes into consideration previous searches, preference profile, location and other attributes.

## Main algorithmic approaches

We can generally divide them by their approach mentioned in [4] and [5] into:

- **Collaborative filtering** (CF)
  Solely based on ratings of items from users (user-item interactions). Trying to recommend unseen items that were liked by users who have a similar taste for other items that they both rated. And thus exploiting data of users with similar preferences.
- **Content-based filtering** (CB)
  Uses item features or item descriptions to recommend items similar to those that the user liked or interacted with. We are essentially building a model of preference for users and exploiting domain knowledge about items that match the users' model.
- **Constraint-based recommendation**
  Depends on hand-crafted deep knowledge about items. User specifies a set of criteria based on which the system filters out items that meet the stated requirements. Additionally the system can sort the items based on their properties, if the stated criteria come with perceived importance - utility.
- **Hybrid systems**
  Combines multiple RS, either the same type with different parameters and/or different types together in order to increase recommendation accuracy. Main types according to [2] are:
  - Weighted where predictions of individual recommenders are summed up.
  - Mixed, where predictions of individual recommenders are combined into one list.
  - Cascade, where predictions of one recommender is feed as an input to another one.

The popularity of the first two approaches varies from domain to domain. Some domains naturally contain item-specific data which allows using the *content-based filtering* for example product parameters in e-shops, but other domains do not. Then it is more beneficial to use *collaborative filtering* techniques or a mix of the two.

There are benefits and drawbacks for both, CF is able to extract latent meaning from the data that would remain inaccessible to CB that relies on items' features. But at the same time, it can cause problems to rely only on user-item interactions because we need a lot of data in order to make a precise recommendation. There will be nothing to recommend if we cannot find similar enough other users that already rated some unseen items. This problem is called a *cold-start problem*.

Further, the third technique, *Constraint-based filtering* requires a deep knowledge which describes items on a higher level and is not very interesting due to the algorithmic simplicity, we will thus not discuss it further.

One additional approach we didn't include in the list is *Critique-based recommendation*. It's popularity is quite low, but still worth mentioning. It acts as sort of a guide through the item space. Where in cycles we show the user items that are distinct in some property (we could say they lie in different areas of the item space) and user either accepts or rejects them. Based on this feedback we narrow the user's preferences and offer additional set of items that reflect that and try to further guide the user to a satisfactory result.

Some of the classical and more advanced methods include:

- User-based and item-based nearest neighbor similarity [6][7][8]
- Matrix Factorisation techniques[9]
- Deep Collaborative filtering [10][11][12]
- Deep Content extraction[12]

## 2.2 Group recommender systems

### Introduction

So far we have discussed only recommender systems, where the object of a recommendation is a single user. But what do we do, when we have a group of users that we want to recommend to? For example, a group of friends selecting a movie that they want to watch together or a group listening to music?

Group recommender systems (group RS) are an interesting subarea of recommender systems, where the object of a recommendation is not just a single user but multiple individuals forming a group. Results of a recommendation for the group do need to reflect and balance individuals' preferences among all members.

### Challenges

- **How to merge individual preferences**
  The main problem when extending RS systems to support the group setting is how to combine preferences of individual users together. It is possible to not support groups at all and let users deal with the act of combining them via discussion. But then the problem collapses back to single user setting, where the user is the whole group. Therefore we need to decide how and when to merge them. Main two approaches are mentioned bellow in Common approaches.

- **Divergent group preferences**
  Even in single variant systems there exist users so called *Grey-sheep* and *Black-sheep*, these users are hard to recommend to, because their preferences do not align with many or any other users (respectively). This problem is especially hard to solve in Collaborative filtering, which directly relies on finding similarity between users. And the same problem arises in the group setting, where it becomes much harder to find solutions that would be satisfactory to all of its members. So in Group RS the problem of outlining users can be observed on two levels, in the usual situation, where the groups aggregated preferences are outlined and on another level where the inter-group preferences of individual users do not match.

- **Feedback gathering**
  In most applications feedback is gathered directly as well as indirectly. Directly by users rating recommended items, and indirectly by observing users behavior such as which items they've visited or how long they've interacted with the item. Gathering direct feedback in the group setting is still possible, even if harder due to possibility that not all members leave a rating, but gathering the indirect, implicit feedback can become even impossible, depending on how the system-user interactions are designed. In most cases users will be selecting an item on a single device, under account of one person, therefore it is hard to distinguish what are preferences of that one individual and what are preferences of the group.

- **Active/passive, primary/secondary group members**
  Another interesting issue arises when we consider that possibly not all members are equally important when it comes to the recommendation. One example could be when parents select a movie to watch with their children, the children should arguably be given a priority over the selection. Second example could be that we would possibly want to prioritize satisfaction of individual in the group that were less satisfied the last time an item was consumed.

## Common approaches

Now we will introduce the two main algorithmic approaches of group recommender systems, according to [13] these are:

- **Group aware RS approach**
  Builds a model for the group based on the preferences of all of its members. Either directly by creating a model of preference for the group or by aggregating models of individual users together and then recommending items for the group as a single entity.

- **RS aggregation approach**
  Use single-user RS to recommend to each individual of the group and then aggregate the results together to create the final recommendation for the group.

In the RS aggregation approach we further distinguish between situations where we have predictions for all possible items and therefore can do aggregation directly on the ratings of all items or if only have a list of recommendation for each user - subset of all the items. These two can function very differently, for example taking in context only the position in the recommended list or position and the rating. They are mentioned separately in [13], but the approaches are very similar, they only differ in the availability of provided results from the underlying RS, so we group them together under one main direction.

Further, both group aware RS and aggregation approaches do both have some advantages and disadvantages. One of advantages of the Aggregation approach is that we can use the same RS as we would use for an individual recommendation, either as a black box aggregating directly the top items provided, or in more involved way by utilizing the predicted ratings. On the other hand, the aggregation strategies do rely on single-user RS so there is not much that can be done in order to extract some hidden latent preferences of the group, which in case of the first method, the group aware approach, can potentially be extracted.

We will go in-depth to discuss techniques used in the latest literature in chapter 4.

At the same time, we need to define what does it even mean to recommend something to a group. Do we measure it by fairness, overall user satisfaction, or by the least satisfied member of the group? We will go in depth to describe common approaches to these problems in chapter 3.

# 3. Fairness

## 3.1 General

## 3.2 Long-term fairness

## 3.3 Evaluation

# 4. Related work

In this chapter, we will go in-depth discussing common approaches that are being used in the group recommendation task. We introduce the main differences between basic types and go through simple as well as more advanced methods.

We assume that we have individual user preferences (if group preferences are available, then the task actually becomes a simple recommendation with the group acting as a single user). Therefore it is necessary to make a distinction based on where the algorithm goes from the preference of the group's individual users to a result for the whole group.

- **Aggregate models**
  The aggregation works on merging the preferences of each member of the group into a single set of preferences that can be then directly consumed by a recommender system, therefore creating a group preference model. Aggregating the single-user preferences either directly by aggregating ratings of seen, or and rated items, or by aggregating the extracted models of user preference to create a single model for the whole group, such as preference matrix in matrix factorization approaches, text descriptions, or item-based recommendations and so on, we will discuss these techniques later.

  This aggregation step precedes the recommendation step. We can see a visualization in figure 4.1.

- **Aggregate predictions**
  Aggregates predictions outputted from RS. Recommender recommends separately for each member of the group based solely on their single-user preferences. Then the resulting recommended items are aggregated together to a single list of recommendations for the whole group. There are two main ways how the final list can be created. Either directly take items each user likes and append them together in some specified manner, or calculate some common utility function from all recommended items and select those that are the most fitting to the group based on this utility function. We will discuss both in more detail in the section 4.1.

  The aggregation step follows after the actual recommendation step. We can see a visualization of this approach in figure 4.2.

- **Aggregation is an uniform part of the recommender**
  In this case, the algorithm directly works with users of the group and does not allow for a clear distinction of the aggregation step. It is deeply and inseparably built into the algorithm itself. Sometimes the perception of the inseparability of these two steps can vary in the literature. Some could say that for example aggregating the user profiles in matrix factorization makes the aggregation inseparable, because there is a specific reprocessing done before the aggregation step. But others will point out, that the user latent matrix is just a representation of a user preference, even if processed by the algorithm itself. We will let the reader decide for themselves where they see

the distinguishing border. We will briefly discuss the available methods in section 4.2

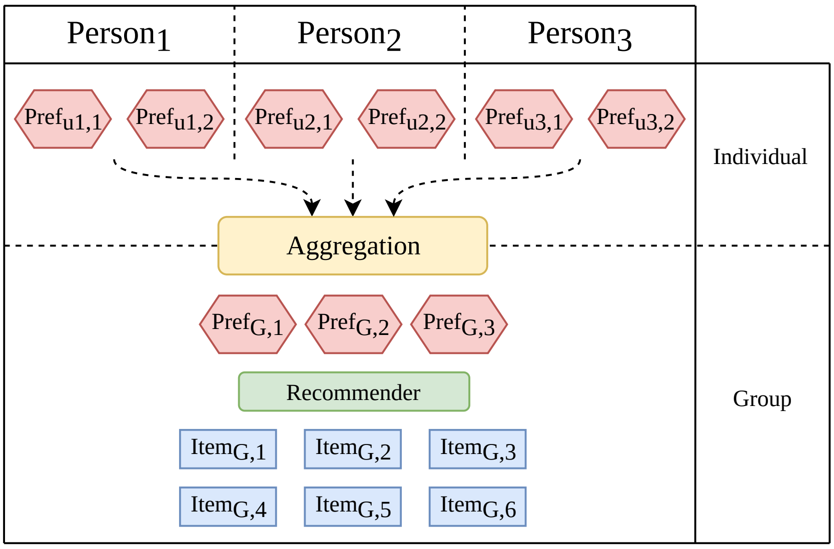


Figure 4.1: High level overview of group recommendation with aggregation of individuals' preferences, before recommendation.

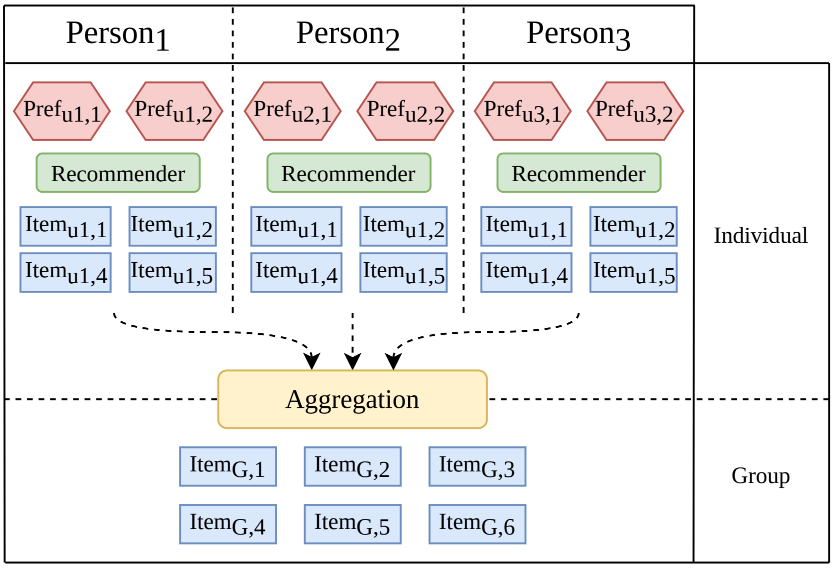


Figure 4.2: High level overview of group recommendation with aggregation on top of recommendation results for individual users.

## 4.1 Simple aggregation methods

We will now introduce the main aggregation functions used, together with an overview of how they interact with the single recommender systems introduced in chapter 2.1

### 4.1.1 Methods

The common aggregation methods as mentioned in [2], [14] and [15] are:

- **Additive utilitarian** (ADD)
  Sum of scores for an item across the group

$$\operatorname*{argmax}_{i \in I} \sum_{u \in G} \mathrm{score}(u, i) \tag{4.1}$$

- **Approval Voting** (APP)
  Number users that like the item above certain threshold

$$\operatorname*{argmax}_{i \in I} \Big| \{ u \in G : \mathrm{score}(u, i) \geq \mathrm{treshold} \} \Big| \tag{4.2}$$

- **Average** (AVG)
  Average of scores for an item across the group

$$\operatorname*{argmax}_{i \in I} \frac{\sum_{u \in G} \mathrm{score}(u, i)}{|G|} \tag{4.3}$$

- **Average without Misery** (AVM)
  Average of scores for an item across the group only if item is above certain threshold for all group members

$$\operatorname*{argmax}_{i \in I : \nexists u \in G | \mathrm{score}(u,i) \leq \mathrm{treshold}} \frac{\sum_{u \in G} \mathrm{score}(u, i)}{|G|} \tag{4.4}$$

- **Borda count** (BRC)
  Sum of scores derived from item rankings. Ranking score is defined for each user by ordering user's items by score and awarding points corresponding to the items location in this ordered list. Worst item receiving 1 point and best item $|I|$ points.

$$\operatorname*{argmax}_{i \in I} \left( \sum_{u \in G} \mathrm{RankingScore}(u, i) \right) \tag{4.5}$$

  Where *ranking score* is defined as follows:

$$\mathrm{RankingScore}(u, i) := \Big| \{ i_{other} \in I : \mathrm{score}(u, i_{other}) \leq \mathrm{score}(u, i) \} \Big|$$

- **Copeland rule** (COP)
  Difference between number of wins and looses for pair-wise comparison of all items

$$\operatorname*{argmax}_{i \in I} \Big( \mathrm{W}(t, I - t) - \mathrm{L}(t, I - t) \Big) \tag{4.6}$$

- **Fairness** (FAI)
  Users in turn, one after another select their top item.

$$\operatorname*{argmax}_{i \in I} \mathrm{score}(u_{current}, i) \tag{4.7}$$

  Where $u_{current}$ is user selected from $G$ for each iteration according to some (in most cases circular, or ping pong) rule.

- **Least misery** (LMS)
  Uses the lowest received rating among the group members as the item's aggregated rating.
  $$\underset{i \in I}{\operatorname{argmax}} \left( \min_{u \in G} (\text{score}(u, i)) \right) \tag{4.8}$$

- **Most Pleasure** (MPL)
  Uses the highest received rating among the group members as the item rating.
  $$\underset{i \in I}{\operatorname{argmax}} \left( \max_{u \in G} (\text{score}(u, i)) \right) \tag{4.9}$$

- **Majority Voting** (MAJ)
  Uses the rating that was given by the majority of the group's members. (Can only work on discrete ratings)
  $$\underset{i \in I}{\operatorname{argmax}} \left( \underset{u \in G}{\operatorname{mode}} (\text{score}(u, i)) \right) \tag{4.10}$$

- **Most Respected Person** (MRP)
  Uses rating proposed by the most respected member of the group.
  $$\underset{i \in I}{\operatorname{argmax}}\, \text{score}(u_{most\_respected}, i) \tag{4.11}$$

- **Multiplicative** (MUL)
  Multiplies all received ratings together.
  $$\underset{i \in I}{\operatorname{argmax}} \left( \prod_{u \in G} \text{score}(u, i) \right) \tag{4.12}$$

- **Plurality Voting** (PLU)
  Each user has a set number of votes that get distributed. The item with the most received votes is selected.
  $$\underset{i \in I}{\operatorname{argmax}} \left( \sum_{u \in G} \text{VotesAwarded}(u, i) \right) \tag{4.13}$$

  Where *votes awarded* is some function that decides for each user how the available votes will be distributed among the items.

### 4.1.2 Usage

## 4.2 Advanced aggregation methods

### 4.2.1 GFAR

## 4.3 Direct model methods

# 5. Datasets

## 5.1 Main datasets

## 5.2 Group datasets

## 5.3 Creating of artificial groups

# 6. Our work

## 6.1 EP-FuzzyD'Hondt

# 7. Offline experiments

## 7.1 Our work

## 7.2 Proceedings

# 8. Application

## 8.1 Design requirements

## 8.2 Architecture and design choices

## 8.3 Cold start problem

## 8.4 User manual

# 9. User study

## 9.1 Methodology

## 9.2 Results

## 9.3 Discussion

# 10. Conclusion

# Future work

# Bibliography


[1] Wikipedia. Recommender system — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Recommender%20system&oldid=1020619015`, 2021. [Online; accessed 01-May-2021].

[2] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. *Group recommender systems: An introduction*. Springer, 2018.

[3] Bashir Rastegarpanah, Krishna P Gummadi, and Mark Crovella. Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 231–239, 2019.

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[5] Alexander Felfernig and Robin Burke. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, pages 1–10, 2008.

[6] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, 1995.

[7] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[8] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[11] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[12] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[13] Anthony Jameson and Barry Smyth. Recommendation to groups. In *The adaptive web*, pages 596–627. Springer, 2007.

[14] Judith Masthoff. Group recommender systems: Combining individual models. In *Recommender systems handbook*, pages 677–702. Springer, 2011.

[15] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. In *Personalized digital television*, pages 93–141. Springer, 2004.

# List of Figures

# List of Tables