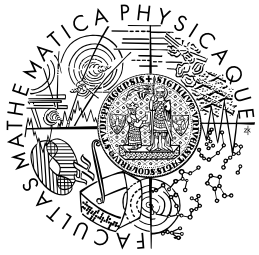

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

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.

2.1.1 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.

2.1.2 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

2.2.1 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.

2.2.2 Characteristics of group RS

2.2.3 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 as mentioned in [13]. 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.

2.2.4 Common approaches

Now we will introduce the two main algorithmic approaches of group recommender systems, according to [14] 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 [14], 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

So far, we have discussed recommender systems and the methods that are being used in the field. Now we will step back and look at the problem from a broader perspective. Specifically, we want to focus on the importance of fairness and to see how and if we can make group recommenders better if we understand it and define it properly.

We will start with general introduction of the topic of fairness. Define its possible meanings and specify which one is important in our setting. This is required due to the overload of the word it self and the rising importance of the topic in today's world. Further, we will explain why fairness seems to be very important parameter in the group recommender setting and will try to reason about how to measure its effects.

Plan:
- Introduce fairness in general
- Go through possible meanings
- Why it's gaining importance and study focus
- Target the one specific meaning we are focusing on

3.1 General

The word it self is defined in Cambridge Dictionary [**?**] as "The quality of treating people equally or in a way that is reasonable." And its use have been rising steadily from the 1960s as we can see on

We, humans, are obsessed with fairness. From young age, children will get sad when something is not fair. When their sibling gets a bigger piece of a pie, more attention from their parent, or any other unequal situation. It can induce strong emotions such as envy, sadness or anger. And this is not limited only to humans, we can observe the same behavior in monkeys. In [15], the authors observed that if a monkey is getting worse reward for the same task as their peer, then it refuses the reward and demands the same payout. Even if they were satisfied with the lesser reward for the same task before.

The next paragraph is a lot of information that doesn't fit well together in small amount of text and should probably be rewritten to become more readable.

But it does not hold for other species of the animal kingdom in general. It seems that it requires a certain level of intelligence in order for the notion of fairness to emerge. As discussed in [16], based on studies of other non-human species, this evolutionary puzzle can be further dissected into responses to reward distribution among the cooperating group members. Where humans are even willing to seek equalization of outcomes even if it means that they will loose some of their own reward as studied in [17]. At the same time we as humans like "free-rides" where us personally get high reward for small amount of work but dislike when someone else gets the same. Which directly corresponds with the fairness it self - "It is not fair, if someone else gets something we don't". But that all depends on our personality and type of relationship that is involved.

It could even be possible, that the notion of fairness which emerges together with cooperation and therefore language/communication is an inherent property of any intelligent agent that is created through evolutionary process.
Let us now get back to fairness in the context of computer systems.

3.1.1 Possible meanings and definitions

We will focus on fairness in regards of society or individuals interacting with a computer system and we will not discuss further any meaning of fairness outside of the domain of computer science.

Most of the recent literature concerning impacts of algorithms to society, fairness is studied from the perspective of discrimination. We can observe some bias to certain groups of population based on their race, sex, or other factors. These biases are introduced directly from the data the algorithms are trained on. And it is important to study techniques and strategies to mitigate this unfairness.

Further, we can also view fairness from the aspect of algorithmic decision making, where a decision process can introduce unfairness based on some non-deterministic property or computation. Some sectors such as justice and finance have to strive for equality of outcome due to possibly of high cost of errors/unfair decisions, either in the form of unjust punishments in the former case or financial loss in the latter one.

Next possible interpretation that applies to our group recommender domain is the notion of fairness in the sense of balance of preference between members of some group. Each member has their own preference and we are trying to balance them in a best possible way so that everyone likes the recommended object or list of objects equally. They may like the recommendation less due to their favorite items maybe not being present or even to like it more due to some introduced novelty from the members having a different preference.

3.2 Application in Group recommender systems

We will now focus only on the group RS setting. We will now mention some of the main ways how fairness can be understood in this context:

- **Fairness distribution in single recommendation**
  We are recommending a single list or a single item and have to balance the items that all members will be satisfied by the selection. We can view fairness in this setting as an optimization problem, where items are considered better if they are liked on average (among the group members) more that items that are liked by some part of the group and disliked by the other. As the group size grows this is harder and harder to do, because a bigger group will most probably have wider preference spectrum which will be harder to satisfy.

- **Fairness in list of items that are consumed sequentially** This setting differs from the previous by the fact that we can be recommending items that are less universally likeable but more specific and liked by part of the group. Therefore intertwining the items in a way that each member will be

satisfied "at some point". The balance of how group-likeable or member/s-likeable items are can be set according to the deployment of the system.

- **Long-term fairness**
  Further, we can distinguish another specific case where recommendations are provided in batches that are separated by some bigger amount of time (days or longer). This case is somewhat similar to the last one, but differs by the fact that unfairness can be more costly to repair. If a person is unsatisfied while watching a movie that was recommended by a group RS, they will less likely to be part of the recommendation in the future. So the balance mentioned in the previous setting is even more important and sensitive parameter to tune. And at the same time it is more important to gather and process feedback.

- **Uneven importance of the group members** In some cases, there will be a situation where the expectation of fairness is distributed nonuniformly. For example when watching a movie with your kids, you probably care about the satisfaction of the kids than the parents. But at the same time, you want to take them into account. In these cases it is important to view fairness as a fluid parameter that can be modified and satisfied by an uneven criteria towards group members.

-

3.3 Other properties

3.4 Evaluation

4. Related work

In this chapter, we will go in-depth discussing common approaches that are being used in the group recommendation task. First, specifying the main approaches, dividing them into groups by how they approach the transition from single-user preferences to group results. Then mentioning the main simple techniques and describing how they interact with the common recommendation approaches. And at last diving into some of the more advanced methods that optimize the aggregation in a more elaborate way.

4.1 Categories

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.

We can put each algorithm into one of the following three groups based on the aggregation step:

- **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 recommended to each user and append them together in some specified manner, or the second way, 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 4.2.

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

- **Aggregation is a 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 preprocessing 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 4.3

This presented grouping based on the aggregation step is different from the main literature [14] and [2], where they also make a distinction into three groups, but instead based on the data that the aggregation processes and the position. Instead we have chosen a little different distinction based more on interaction of the aggregation with the recommender system, essentially putting two groups from aforementioned literature under single group (*aggregate models*), but with the mentioned possible subdivision.

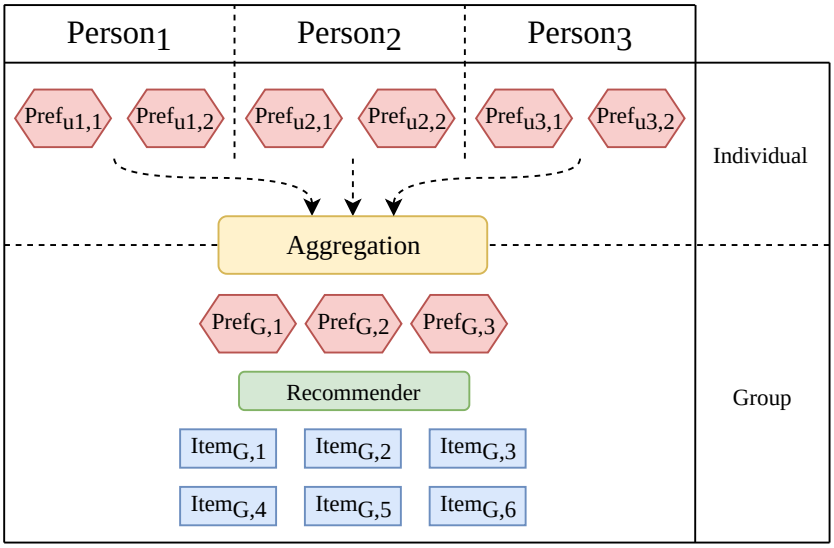

Figure 4.1: High level overview of group recommendation with aggregation of individuals' preferences, before recommendation.

4.2 Simple aggregation methods

We will now introduce the main aggregation functions (interchangeably as "aggregation strategies", "aggregation methods") used, together with an overview of how they interact with the single-user recommender systems introduced in chapter 2.1.2

4.2.1 Methods

The main aggregation methods can be divided into three groups, based on their high-level approach into - majority, consensus, and borderline based strategies.

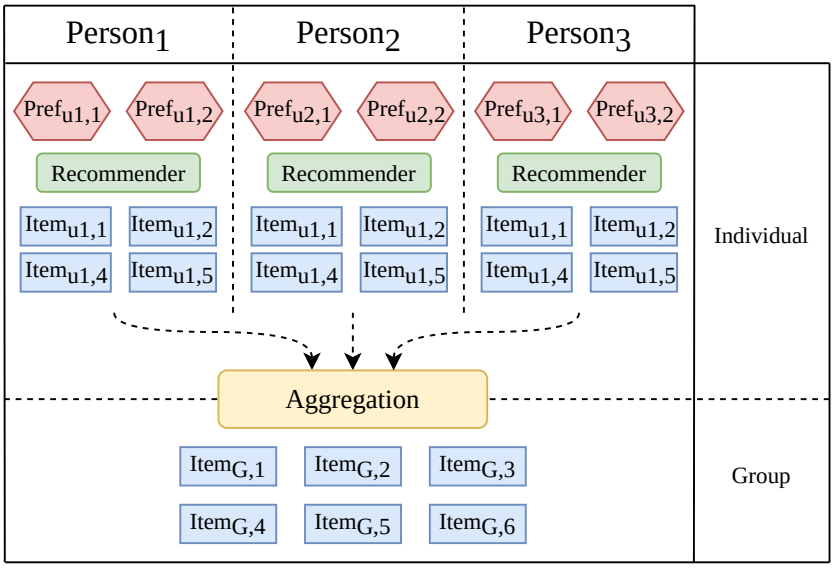

Figure 4.2: High level overview of group recommendation with aggregation on top of recommendation results for individual users.

The majority-based generally use the most popular item among the group members, the consensus-based consider preferences of all the group members, and the borderline-based consider a subset of the items by some limiting criteria.

We now list the most common aggregation methods as mentioned in [2], [18] and [19] specify to which of the group mentioned above they belong.

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

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

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

$$\underset{i \in I}{\operatorname{argmax}} \left| \{u \in G : \operatorname{score}(u, i) \geq \operatorname{treshold}\} \right| \tag{4.2}$$

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

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

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

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

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

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

  Where *ranking score* is defined as follows:

$$\operatorname{RankingScore}(u, i) := \left| \{ i_{other} \in I : \operatorname{score}(u, i_{other}) \leq \operatorname{score}(u, i) \} \right|$$

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

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

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

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

  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, borderline)
  Uses the lowest received rating among the group members as the item's aggregated rating.

$$\underset{i \in I}{\operatorname{argmax}} \left(\min_{u \in G} \left(\operatorname{score}(u, i) \right) \right) \tag{4.8}$$

- **Most Pleasure** (MPL, borderline)
  Uses the highest received rating among the group members as the item rating.

$$\underset{i \in I}{\operatorname{argmax}} \left(\max_{u \in G} \left(\operatorname{score}(u, i) \right) \right) \tag{4.9}$$

- **Majority Voting** (MAJ, majority)
  Uses the rating that was given by the majority of the group's members. (Can only work on discrete ratings)

$$\underset{i \in I}{\operatorname{argmax}} \left(\operatorname{mode}_{u \in G} \left(\operatorname{score}(u, i) \right) \right) \tag{4.10}$$

- **Most Respected Person** (MRP, borderline)
  Uses rating proposed by the most respected member of the group.

$$\underset{i \in I}{\operatorname{argmax}} \operatorname{score}(u_{most\_respected}, i) \tag{4.11}$$

- **Multiplicative** (MUL, consensus)
  Multiplies all received ratings together.

$$\underset{i \in I}{\operatorname{argmax}} \left(\prod_{u \in G} \operatorname{score}(u, i) \right) \tag{4.12}$$

- **Plurality Voting** (PLU, majority)
  Each user has a set number of votes that get distributed. The item with the most received votes is selected.

$$\underset{i \in I}{\operatorname{argmax}} \left(\sum_{u \in G} \operatorname{VotesAwarded}(u, i) \right) \tag{4.13}$$

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

Some of the methods have an additional distinguishing factor, which is that they are iterative calculations instead of a single calculation that returns the final ordering. Most notably *Fairness*, it calculates the next item purely from a currently selected user, users changing in iterations one after another. Another one is *Plurality Voting*, this technique can also be iterative but it is not mandatory. Where after the top item is selected we reset the calculation, therefore selecting top items in an iterative manner. All the other ones are single-iteration methods, where the final list requires only one calculation.

Now we will show how these strategies work together with a recommendation in order to provide a full group recommender system, which will serve as an introduction of the inner workings of such system in order for us to then continue towards more advanced aggregation methods.

4.2.2 Usage with recommender systems

We now need to distinguish between *model aggregation* and *prediction aggregation* due to in most cases very different inputs and required outputs from the aggregation.

Prediction aggregation

In most cases using them as *prediction aggregation* after the recommendation is quite straightforward. In all (to us known) cases RS returns recommended items together with some sort of rating of the recommendation. There are some applications where this presumption does not hold, such as building a group recommendation aggregation on top of a *black box* recommender that only returns a list of items, for example, if extracting recommended items data from a web page where we only retrieve the list itself and we do not have access to the underlying recommender system. But in the following section we will assume that we have a full access to the rating data of the outputted items. We further assume that if necessary we can acquire ratings and ordering for all possible items.

As shown in figure 4.2, we first get a list of recommended items (together with ratings) for each user. Then we can directly apply the methods mentioned in section 4.2.1.

Model aggregation

Running aggregation on top of user preferences, instead of lists of recommended items is more challenging due to many possible inputs that the recommender can take, such as explicit or and implicit feedback. As discussed in section 4.1, we have two main options, aggregation of ratings and aggregation of mined preferences.

First, the aggregation of ratings, in other words creating a set of ratings that represent the whole group. This approach is preferred due to its simplicity. We can use many of the before mentioned simple methods, mainly the *consensus-based* which aggregate ratings directly (positive as well as negative ratings). Other mentioned methods can be used as well, but some of them do not make much sense to adapt, for example, the voting-based, with which we will be very limited due to the usual sparsity of the feedback ratings that we have available.

The second option, the aggregation of mined preferences, is more present in content-based systems. Where we aggregate not the rating themselves but some other data that represent the users' likes and dislikes. As an example, we will use the *tf-idf* keyword extraction which is by far the most popular method used in the text-based recommendation as mentioned in [20]. We can weight *tf-idf* concepts extracted from the user feedback by the received rating, and then aggregate these concepts together for the whole group. As we can see, it is a little more elaborate approach that requires more modifications to the simple methods mentioned.

4.3 Advanced methods

As we can see, the simple methods mentioned before are quite straightforward, they have mostly very simple and clear objectives, and try to optimize in different ways some quite vaguely specified goal. But as we have discussed in the chapter 3, the objective is quite hard to define and measure. We will now introduce some other methods that either directly or indirectly try to optimize some better-specified goal.

4.3.1 GFAR

So far we have introduced methods that do not directly specify any form of optimization in a direction of *fairness*. A new method introduced in[21] directly optimizes the resulting list to balance the relevance of the recommended items in a rank-sensitive way.

Introduction

As mentioned they focus on the *fairness* of top-N_G (*fairness* of the top N recommended items for the group G). Where they define *fairness* as a property of the top-N_G items, not just any single item but the whole list. As already discussed in chapter 3.2 the difference between optimization independently item after item versus in some way for the whole list can yield significantly better results when it comes to sequential consumption of items or and balancing users with different preferences. For example, being one member of the group being treated unfairly due to different preferences from the rest of the group, then fairness defined as

a property of the whole list can in some way compensate and give more priority for balancing fairness forwards this one user. This kind of unfairness is the main problem that they try to solve.

We have already mentioned one algorithm that tries to balance fairness of the whole list in chapter 4.2.1, FAI, which generates the aggregated results in iterations, each iteration optimizing for a different user in turns. This approach solves the aforementioned problem in a very simple (although somewhat dull) way by not considering the group preferences at all and just trying to please everyone in turns not considering if the item will be liked by the rest of the group.

Approach

What if we have instead taken into account the previously recommended items when we are selecting the next one in the list? The authors propose defining *fairness* together with ordering within the group by saying that "a top-N is fair to a group if the relevance of the items is balanced across the group members for *each prefix* of the top-N_G". In other words for each iteration, we try to select an item that improves the balance as much as possible.

Let us first define Borda relevance following [22] as:

$$\text{Borda-rel}(u,i) = \left|\left\{j : \text{rank}(j, \text{top-}N_u) < \text{rank}(i, \text{top-}N_u)\right\}, \forall_j \in \text{top-}N_u)\right| \quad (4.14)$$

Where rank is the position of item i in the user's u candidate list (position is determined by ordering the items based on the returned items score, from the best to the worst). Borda relevance essentially gives zero points to the last item and increases the points by one for each position up in the list, with the first/top item getting $(\#\text{items} - 1)$ points.

Now we can set probability of item i being relevant for user u as:

$$p(\text{rel}|u,i) = \frac{\text{Borda-rel}(u,i)}{\sum_{j \in \text{top}-N_u} \text{Borda-rel}(u,j)} \quad (4.15)$$

Let also $p(\neg\text{rel}|u,S)$ be the probability that item i is not relevant for any of the items in set S. We derive the probability that atleast one item from set S is relevant to the user u as:

$$\begin{aligned} p(\text{rel}|u,S) &= 1 - p(\neg\text{rel}|u,S) \\ &= 1 - \prod_{i \in S} (1 - p(\text{rel}|u,i)) \end{aligned} \quad (4.16)$$

Now we want to generalize for the whole group, so we define $f(S)$ as the sum of probabilities for each user that they find at least one relevant item in the set S:

$$\begin{aligned} f(S) &= \sum_{u \in G} (1 - p(\neg\text{rel}|u,S)) \\ &= \sum_{u \in G} \left(1 - \prod_{i \in S} (1 - p(\text{rel}|u,i))\right) \end{aligned} \quad (4.17)$$

We have used the Group G as a constant in equation 4.18 and forward, the group will stay the same during the calculation. $f(S)$ shows how to balance the

fairness amongst the group members for a specified set of item. We now want to extend it in order to make it rank-sensitive by defining a marginal gain in function f for adding a new item i to the set S as follows:

$$\begin{aligned} f(i, S) &= f(S \cup i) - f(S) \\ &= \sum_{u \in G} \left[p(\text{rel}|u, i) \prod_{j \in S} (1 - p(\text{rel}|u, j)) \right] \end{aligned} \tag{4.18}$$

Where we have used equations 4.16 and 4.18. And finally, we can define an ordered set that is considered fair if it balances each of its prefixes. In other words, the first item of the set should be considered fair/balanced by all group members as much as possible, then the first two items and so on, we define fairness in an ordered set OS as:

$$\text{fair}(OS) = \sum_{k=1}^{|OS|} f(OS)[k], \{i \in OS : \text{rank}(i, OS) < k\}) \tag{4.19}$$

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] Luis M de Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Managing uncertainty in group recommending processes. *User Modeling and User-Adapted Interaction*, 19(3):207–242, 2009.

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

[15] Sarah F Brosnan and Frans BM De Waal. Monkeys reject unequal pay. *Nature*, 425(6955):297–299, 2003.

[16] Sarah F Brosnan and Frans BM de Waal. Evolution of responses to (un) fairness. *Science*, 346(6207), 2014.

[17] Daniel John Zizzo and Andrew J Oswald. Are people willing to pay to reduce others' incomes? *Annales d'Economie et de Statistique*, pages 39–65, 2001.

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

[19] 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.

[20] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[21] Mesut Kaya, Derek Bridge, and Nava Tintarev. Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In *Fourteenth ACM Conference on Recommender Systems*, pages 101–110, 2020.

[22] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. Fairness-aware group recommendation with pareto-efficiency. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 107–115, 2017.

List of Figures

List of Tables