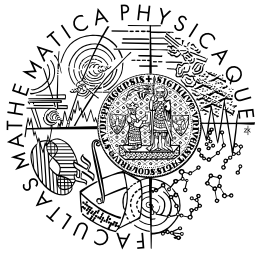

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

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 **3**
- 1.1 Problem statement . . . 3
- 1.2 Research objective . . . 4
- 1.3 Thesis structure . . . 4

2 Recommender systems **5**
- 2.1 Recommender systems . . . 5
  - 2.1.1 High-level examples . . . 5
  - 2.1.2 Main algorithmic approaches . . . 6
- 2.2 Group recommender systems . . . 7
  - 2.2.1 Introduction . . . 7
  - 2.2.2 Characteristics of group RS . . . 7
  - 2.2.3 Challenges . . . 8
  - 2.2.4 Classification . . . 9
  - 2.2.5 Common approaches . . . 9

3 Fairness **11**
- 3.1 General . . . 11
- 3.2 Algorithmic fairness . . . 13
  - 3.2.1 Sources of algorithmic unfairness . . . 13
  - 3.2.2 Examples of algorithmic bias . . . 14
  - 3.2.3 Measures of algorithmic fairness . . . 15
  - 3.2.4 Outcome vs. opportunity . . . 16
  - 3.2.5 General methods of prevention . . . 17
  - 3.2.6 Other adverse effects . . . 18
- 3.3 Fairness in Group recommender systems . . . 19
  - 3.3.1 Specific cases of fairness . . . 19
  - 3.3.2 Evaluation . . . 20
- 3.4 Other criteria . . . 20
  - 3.4.1 Bias . . . 21
  - 3.4.2 Privacy . . . 21
  - 3.4.3 Trust . . . 22
  - 3.4.4 Explainability . . . 22
  - 3.4.5 Legitimacy . . . 22
  - 3.4.6 Consistency . . . 24
  - 3.4.7 Novelty . . . 24
  - 3.4.8 Coverage . . . 25
  - 3.4.9 Diversity and similarity . . . 25
  - 3.4.10 Accuracy and precision . . . 25
  - 3.4.11 Performance . . . 25

4 Related work **26**
- 4.1 Categories . . . 26
- 4.2 Simple aggregation methods . . . 27
  - 4.2.1 Methods . . . 27

4.2.2 Usage with recommender systems . . . 30
4.3 Advanced methods . . . 31
4.3.1 GFAR . . . 31
4.3.2 Package-to-Group Recommendations . . . 33
4.3.3 Top-N group RS with fairness . . . 33

5 Datasets **34**
5.1 Main datasets . . . 34
5.2 Group datasets . . . 34
5.3 Creating of artificial groups . . . 34

6 Our work **35**
6.1 EP-FuzzyD'Hondt . . . 35

7 Offline experiments **36**
7.1 Our work . . . 36
7.2 Proceedings . . . 36

8 Application **37**
8.1 Design requirements . . . 37
8.2 Architecture and design choices . . . 37
8.3 Cold start problem . . . 37
8.4 User manual . . . 37

9 User study **38**
9.1 Methodology . . . 38
9.2 Results . . . 38
9.3 Discussion . . . 38

10 Conclusion **39**

Conclusion **40**

Bibliography **41**

List of Figures **45**

List of Tables **46**

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. Recommender systems has become quite literally 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 as widely used as the non-group variants, because we mostly use the web, listen to music and read the news as individuals. At least from the perspective of those systems. However, 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.

With groups as the target of a recommendation comes new challenges, among many others, how to measure satisfaction and ensure fairness among the group members. We first need to have a reliable way how to evaluate the recommendations. 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 with metrics other than the classical accuracy and precision) thinking about them is quite recent. With the rise of social dilemmas around recommender systems is the fairness-ensuring topic becoming more important than before. 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 are utilized the most.

- **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 feedback 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 efficacy. 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 did not 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 critique from user we narrow down the user's preferences and offer different (filtered/extended) set of items and try to further guide the user to a satisfactory result. The critique can some cases be provided for not just the whole item but even just the properties or part of the items. An example could be: 'This carpet has a beautiful pattern but the color is not that nice.'

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 (hereinafter referred to as single user RS). 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

mention some basics characteristics, how many recommended, feedback gathering, timeframe

intro into next subchapters

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 in 2.2.5.

- **Divergent group preferences**
  There exist users that are 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 outlying 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 explicitly as well as implicitly Explicitly by users rating recommended items, and implicitly by the system observing users' behavior such as which items they have visited or how long they have interacted with the item. Gathering direct feedback in the group setting is still possible, even if it is harder due to the possibility that not all members leave a rating. In some cases 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.

- **How to explain provided recommendations**

  Unfinished

- **Not all members present**

Unfinished

- **Selecting from the provided list**

  Unfinished

2.2.4 Classification

As found in [14]

Unfinished

- pref known or gathered overtime
- just presented or experienced in real time
- group is passive or active
- recommending single item, set, list, sequence?

  We need to distinguish what the result will be used for, generating the recommendations for a list where a single item will be selected is different from when we generate a list that will be consumed in item after item fashion. Is there a naming for this? Single item vs K-item utility?? [15] and the original paper [16]

2.2.5 Common approaches

Mention ephemeral vs persistent from paper PolyLens: A Recommender System for Groups of Users

Now we will introduce the two main algorithmic approaches of group recommender systems, according to [17] 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.

Mention division into groups based on where the recommender is "located" in relation to the aggregation, in short and then ref. the reader to the related work chapter where we discuss in depth. Or move the division from the related work here?

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 [17], 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 used in the field. But now, let us step back and look at the problem from a broader perspective of fairness as a social construct. Specifically, we will focus on the importance of fairness in the context of algorithms, what role it plays when using machine learning models with potentially sensitive data and see how and if we can make group recommenders better when we understand and define the underlying properties.

We will start with a general introduction to 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 itself and the rising importance of the topic in today's world. Furthermore , we will explain why fairness seems to be a crucial parameter in the group recommender setting and will try to reason about how to measure its effects.

3.1 General

The word fairness itself is very hard and controversial to define. In Cambridge Dictionary [18] it is defined as "The quality of treating people equally or in a way that is reasonable." Its use has been rising steadily from the 1960s as we can see on 3.1.

Humans are obsessed with fairness. From a young age, children will get sad when something is not fair, when their sibling gets a more significant 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 it emerges very early, as early as 12 months of age or sooner as researched in [20].

And this behavior is not limited only to humans. We can observe the same behavior in monkeys. In [21], the authors observed that if a monkey is getting the worse reward for the same task as its peer, it refuses the reward and demands the same payout even if they were satisfied with the lesser reward for the same task before. In some sense, this is very strange, why should you care about someone having more if you have enough?

We observe the same behavior in many other species of animals, but not all. It seems that it requires a certain level of intelligence for the notion of fairness to emerge. As discussed in [22], 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 willing to seek equalization of outcomes even if it means that they will loose some of their own reward as studied in [23]. At the same time, we humans like "free-rides" where we get high rewards for a small amount of work but dislike when someone else gets the same. This directly corresponds with the fairness itself - "It is not fair if someone else gets something we don't.". However, it all depends on our personality and the type of relationship involved. Some people are, for example, willing to accept that their partner makes more, but that is very subjective and in some cases can lead even to larger envy involved.

In some cases we observe a pattern of generosity, where children are willing

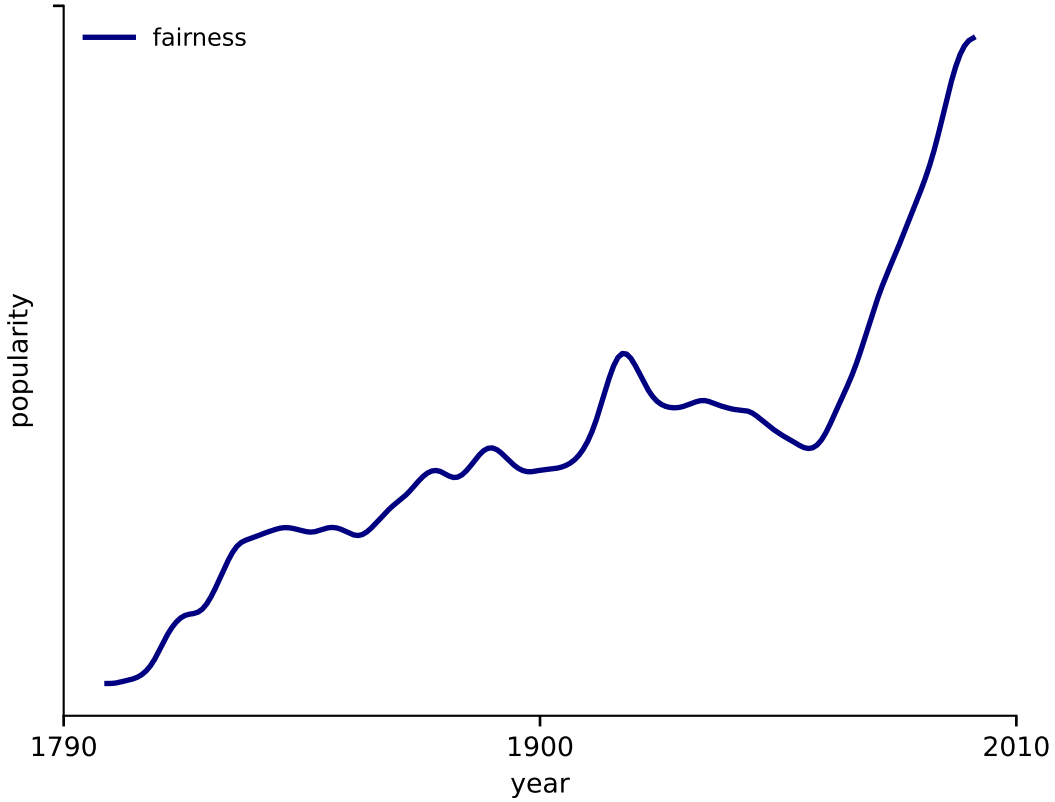

Figure 3.1: The graph shows the phrase ”fairness” occurring in the corpus of English books from 1800 to 2010. Source: Google Ngram Viewer, corpora 2012. [19]

to make their own sacrifices in order to ensure fairness, so that the other person does not have less than them, as presented in [24]. The authors studied fairness in multiple cultural settings and found out that this behavior is learned and only present in some cultures.

On the other hand, our society widely accepts the notion of "winner takes all", which can be problematic in itself, for example, in sports and business. In business it directly shapes the distribution of wealth, which is considered as one of the main problems of today's world. But discussing the social aspect of fairness is beyond the scope of this thesis.

When it comes to the true nature of fairness on the deepest level, it could even be possible that the notion of fairness emerges together with cooperation and therefore language and communication. It would mean that it is an inherent property of any intelligent agent that is created through an evolutionary process. Another point of view could be that fairness acts as a mechanism that pushes towards equality among the group members and that leads to a higher stability of the group which would give an evolutionary advantage. But more research has to be done, as our understanding of intelligence, consciousness, and related hard to quantify phenomena is lacking.

Let us now get back to fairness in the context of computer systems.

3.2 Algorithmic fairness

We will focus on fairness regarding society or individuals interacting with a computer system. We will not discuss further any meaning of fairness outside of the domain of computer science. This topic steers away from the main goal of defining fairness for the group RS sub-domain, but we think that it is very important topic in general, deserves more spotlight and can be a good middle step to understand specifics of the group RS setting.

3.2.1 Sources of algorithmic unfairness

How can a computer that has no underlying understanding of race or ethnicity discriminate against a group of people? At first this idea may seem strange, but we have to remember, that as with any other computer program, machine learning algorithms are designed by people. Data that is used to train those algorithms come from the real world, where bias and unfairness is unfortunately still present. And so, the trained models even if not usually meant with an ill-fated purpose will reflect that and in most cases include some form of bias or unfairness. Of course, there exist uses of ML with bias that has been introduced on purpose. We can see that for example in the Chinese credit score system which is biased towards an individual based their class, race, political views and other factors which we, in the modern democratic world, consider as protected(sensitive) characteristics, which are not to be used as discrimination against an individual. But this type of bias in those circumstances is a knowingly designed and required feature of the system and therefore we will not discuss these systems any further.

In a more rigorous way, we can say that output of any machine learning (ML) algorithm is usually just a product of the underlying data. By design, accurate

classifier will reproduce patterns found in the training data. And usually the bias is either transferred directly from the data or by wrongly defining the learning objective.

Let us now present a division by the main sources of unfairness as stated in [25] with examples following in 3.2.2:

- **Biases already included in the data set**
  Such as dependence/correlation of data based on sensitive characteristics. Bias found in the data is by design reproduced by an accurate classifier.

- **Biases caused by missing data**
  Missing or filtering out some of the training data can results in a data set that does not represent the target population.

- **Biases stemming from algorithmic objectives**
  While training, we usually minimize some error, but that can lead to prioritization of interests of majority if left unchecked. It will always be easier to optimize results for groups with small entropy, than for niche groups that are more surprising and thus having a larger entropy.

- **Biases caused by "proxy" attributes**
  Some attributes that are not directly considered sensitive still can contain information from sensitive attributes. In other words, they are not independent. The algorithm can therefore use the "proxy" attribute and exploit the sensitive attribute indirectly.

It is important to define which sensitive characteristics need to be taken into account. As stated in [26] those are: gender, ethnicity, sexual orientation, disability, and others. Most research in the domain of bias and fairness is, based on our perception, studied from the perspective of discrimination and impacts of algorithms on society. We observe bias to specific groups of population based on their race, sex, nationality, education, beliefs, and many other attributes (protected, as well as unprotected ones) which causes a measurable impact on our everyday life. We are, after all, are more than ever involved and surrounded by technology. It is therefore essential to understand the effects which biased algorithms can have and to study techniques and strategies to mitigate their negative impact.

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 the high cost of errors of unfair decisions, either in the form of unjust punishments in the former case or financial loss in the latter one.

3.2.2 Examples of algorithmic bias

We will now present a few instances of computer systems that have been used where bias towards a sensitive characteristic had a substantial impact.

- **Amazon's automatic recruiting system**
  As reported by [27], in 2018 it was found that new system for hiring people

for Amazon was biased towards women. IT field being mostly male dominated - women represent only around 23% as stated in [28]. Due to this disproportion the algorithm discovered in training data pattern between gender of the candidate and hiring results which lead it to assume that male candidates are preferred before female ones. The algorithm was not told the gender of the candidate, but it inferred it from other data such as university, hobbies and other. This bias is a combination of "bias already included in the data set" and "biases caused by proxy attributes". The company later disbanded the team and left the tool only as a helper tool that works in conjunction with the recruiters, instead of solely automatically.

- **Apple's credit card**
  Apple released its own credit card in 2019. It works as follows: after the sign up, the user receives certain credit limit by the service provider (Goldman Sachs). Some people, as reported in [29], noticed that their wives were assigned smaller credit limit even though their credit score was higher and they only had one shared bank account. In this case, investigation by New York State Department of Financial Services came to a conclusion based on an extensive analysis that no unlawful discrimination against applicants have taken place.

- **COMPAS** - Correctional Offender Management Profiling for Alternative Sanctions
  COMPAS is an algorithm that was used in US justice system to predict the likelihood that a defendant would become a recidivist. Analysis [30] found that black defendants were often predicted as being higher risk than they actually were and on the contrary white defendants were predicted to be less risky than in reality. In case of re-offended, this predicted risk was almost twice as high for blacks compared to whites. They conclude that the tool is very imprecise and does not reflect the true likelihood that it was designed to predict.

In these cases, society and mainly law has to act and protect those that are treated unfairly. European law can act as a good example of what can be done. The general data protection regulation (GDPR) and protection of individuals against algorithmic bias are some of the great and functioning examples.

Details about laws that are in effect in EU and definitions of sensitive characteristics and areas of protection can be found in Handbook on European non-discrimination law [26].

3.2.3 Measures of algorithmic fairness

We need to have a precise way of how to measure bias towards sensitive characteristics in order to design and evaluate algorithms that are taking or should take measures to ensure fairness.

At first sight our idea could be to just remove features that we consider sensitive entirely from the data set, but that will in most cases not suffice due to other features being slightly correlated with the sensitive feature. Correlation with, for example, gender will probably be too small to to be predicted with measurable

accuracy, but this balance can tip over if we put many of these slightly correlated features together. We therefore need to approach this problem more rigorously.

We now present a few statistical methods as mentioned in [31]:

- **Independence** We say that sensitive characteristic Char is independent of a prediction Pred if:

$$P\left(Pred = p|Ch = a\right) = P\left(Pred = p|Char = b\right) \quad \forall p \in Pred \quad \forall a, b \in S \tag{3.1}$$

  Meaning that the probability of the given prediction is the same for two people from different groups with respect to the sensitive characteristic.

- **Separation** We say that random variables $(Pred, Char, Y)$ satisfy separation if the sensitive characteristics Ch are statistically independent to the target value Y given the prediction $Pred$. This relation can be expressed with:

$$\begin{aligned} P\left(Pred = p|Y = q, Char = a\right) = P\left(Pred = p|Y = a, Char = b\right) \\ \forall p \in Pred \quad q \in Y \quad \forall a, b \in Char \end{aligned} \tag{3.2}$$

  Meaning that dependence of prediction result on the sensitive attribute $Char$ can be justified by the attribute $Char$ being dependent on Y.

- **Sufficiency** We say the random variables $(Pred, Char, Y)$ satisfy sufficiency if the sensitive characteristics A are statistically independent to the target value Y given the prediction R. This can be expressed as:

$$\begin{aligned} P\left(Y = q|Pred = p, Char = a\right) = P\left(Y = q|Pred = p, Char = b\right) \\ \forall q \in Y \quad p \in Pred \quad \forall a, b \in Char \end{aligned} \tag{3.3}$$

  We say that $Pred$ satisfies sufficiency i the target variable Y and the sensitive attribute $Char$ are clear from the context.

3.2.4 Outcome vs. opportunity

From separation and sufficiency, we see that they there is only a difference in the direction of the relationship between random variables $Pred$ and Y, we will call them 'equality of opportunity' and 'equality of outcome' respectively.

Let us now present an example of both. We have a model situation where we strive for gender equality in management of our company.

- **Equality of outcome** We would like the resulting distribution to be fair such as there has to be 50% male and 50% female gender representation among our management. If we have more than 50% men we need to fire them and hire only women. It may seem easy, and this is where most of the efforts usually stop, but even just finding what the desired resulting distribution needs to look like is a non-trivial task.

- **Equality of opportunity** In this case, we try to mitigate any bias that could skew the decision of who to hire towards any of the gender. Preferably, we don't want to even propagate the fact about the protected attribute (in this case gender) to the people making the final decision.

Both of these cases/methods have their place but should be used cautiously. They can cause a great deal of fairness equalization when used correctly, but at the same time great deal of harm, while implemented incorrectly.

With the already discussed topics in mind, we can get back and connect the fairness and the group recommendation systems with the subsequent possible interpretation. What applies to our group recommender domain is the notion of fairness in the sense of balance of preference between members of a group. Each member has their preference, and we are trying to balance them in the best possible way so that everyone likes the recommended object or list of objects equally.

Further, if we take group membership as a sensitive attribute of the group and consider it a sensitive attribute, then we want independence in the context of equality of outcome.

3.2.5 General methods of prevention

Thanks to machine learning models being entirely dependent on the data as discussed previously, we can divide the general techniques of bias suppression by where on the data path is the change to the machine learning process made. We divide the general techniques into three categories as follows:

- **Pre-processing** Adjust training data in a way that sensitive characteristics are uncorrelated.

  The main benefit of preprocessing data this way is that if we ensure independence this way then any subsequent deterministic method will transitively also satisfy independence of the sensitive attributes. But as with any data transformation, that change data properties and distributions, we need to be careful as not to hinder the efficacy of the final model.

- **At training time** Design algorithms that set constraints on the optimization process it self.

  This type of change seems to be the hardest to technically and algorithmically implement. We need to have an access to the whole collection of raw data in order to ensure that we are not biased. Further more, we limit ourselves to ML methods that allow this type of constrain modulation. On the other hand main benefit is that we perform the optimization with full information of the task and therefore can potentially gain more utility.

- **Post-processing** Adjust parameters of already learned classifier so as to be uncorrelated with the sensitive attribute.

  This is the least favorable from the theoretical point of view due to us only being able to correct the bias in the way of ensuring equality of outcome which corresponds with the difference between the before mentioned properties of sufficiency and separation. At the same time it makes evaluation of the model performance harder.

3.2.6 Other adverse effects

It takes a lot of time to make data sets and models unbiased, especially so, because bias is included in most of the data sets that come from the real world. Let us now discuss some other machine learning effects that play a big role when combating algorithmic fairness. We now need to take into account the iterative learning of the machine learning models, meaning that we, after putting them in production, retrain the model on data that was affected or directly generated in or by an environment that the models were part of.

Two of the most common problems are the so called 'negative feedback loop' and 'echo chambers'.

Negative feedback loop

A decision-making system that is learning from past data and is retrained in the future on data that were affected by it's decisions (after being utilized in the environment from which we gather the data set) will consume its own (previous versions') decisions as training data. This can lead to amplification of biases that were already present and to further skew the distribution of the underlying data. This effect is called "negative feedback loop".

In order to build robust and fair algorithmic systems, we need to understand when and why this effect emerges. When we look at it from a simpler perspective, where the current system deployed in production is just a set of predictions, then training the next model is just training the previous with additional data that the current model made (the set of predictions from production). In this sense the set of features selected while the current model was trained will correlate with the new data and therefore affect the selected and used features in the new training. This can be very detrimental to performance of the model. And it is not easily fixable due to how data is gathered and how machine learning is iterated.

We will mention an example from [32]. Lets assume that we are building a decision-making algorithm that decides where to put our call-center capacity in order to generate the most profit. In other words, to which telephone numbers from some candidate list to call. And lets again assume that in the prior data the conversion rate of person coming from page X is 5%, from page Y is 2% and from page Z is 1%. The algorithm will naturally be selecting to focus our capacity to X, because it has the biggest conversion rate. Now fast forward some time when we are retraining the model on new data. Due to us preferring the page X, and not calling to people coming from Y and Z our data distribution of conversion rate looks like this: X: 8.5%, Y: 0.5% and Z: 0%. We end up having less data about Y and Z due to us, not calling people coming from those pages. Now we will retrain the model and this bias towards X will reinforce. In this way the performance of the model is decreasing, due to the fact that data no longer following the actual underlying distribution. If we use concepts from Reinforcement learning - we are exploiting, but not exploring.

This negative feedback loop can be and is very detrimental to models' performance and in a wider view even dangerous to society. Which directly leads us to a second adverse effect of echo chambers.

Echo chambers

The feedback loop described in previous section can be observed in effect not only when retraining algorithms, but in social groups as well. People naturally seek out information that reinforce and support their existing views as stated in [33]. Serving content to user that supports their views will therefore lead to higher satisfaction and interaction with the system.

The main problem becomes the training metrics that try to maximize numerous aspects of the system's performance such as the amount of time user spent interacting with the service and return rate of the visitors. We use them as a target for the optimization and that leads to a creation systems that naturally locks its users in so called "echo chambers". While being inside, your own views will be reinforced. This, together with people increasingly consuming social media feed as their main source of news is probably one of the main forces behind the increasing polarization in society as discussed in [34] and [35].

3.3 Fairness in Group recommender systems

So far, we have discussed algorithmic fairness in the context of equality of opportunity, where the main goal is to not discriminate against an individual. The same issues are present in the Group RS domain, but we will focus on equality of outcome more. Our objective is to fairly distribute item recommendation quality between group members, so that everyone is as satisfied as possible and the level of satisfaction among users is as close to uniform as possible.

Let us first introduce two concepts of item preference:

- **Member-likeable**
  We say that a recommended item is member-likeable, if it is chosen with the aim of satisfying a single-group member, or a small subset of a group, more than with the aim of increasing the average.

- **Group-likeable**
  We say that an item is group-likeable, if it is was selected for recommendation with the goal of uniformly satisfying all group members.

This is one of the main optimization decisions we have to make. In some settings, as we will discuss later, will member-likeability a better choice, in others it will be the group-likeability. We can view it as an opposing force. We can either push towards more or less uniformity.

3.3.1 Specific cases of fairness

Let us now mention some of the main ways and their differences, as how fairness in this context can be understood.

- **Fairness distribution in isolated recommendation**
  We recommend a list of items (possibly even a single item) and have to balance the choice of the items so that all members will be satisfied. This single list is an isolated recommendation. We can view fairness in this setting as a direct optimization problem, where items are considered better

if they are liked on average (among the group members) more than items liked by some part of the group and disliked by the other. As the group size grows, this is harder and harder to satisfy because a larger group will most probably have a broader preference which will be harder to to meet due to the differences in members' taste. We can view preference of a group more like a set intersection than set union.

- **Fairness in a list of items that are consumed sequentially** This setting differs from the previous, because we can be recommending items that are less universally likable but more specific and only liked by a part of the group. Therefore intertwining the items so that each member will be satisfied "at some point". The balance of group-likable or member-likable items can be tuned according to our specific requirements.

- **Long-term fairness**
  Further, we can distinguish another case where recommendations are provided in batches separated by some more significant 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 dislikes the item recommended by a group RS, they will less likely be part of the recommendation in the future. So the balance mentioned in the previous setting is an 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 non-uniformly. For example, when watching a movie with your kids, you probably care about the satisfaction of your kids more than your own. But at the same time, you want to take yourself into account too. In these cases, it is essential to view required fairness as a fluid parameter that can be modified and satisfied by uneven criteria towards group members.

3.3.2 Evaluation

DEADLINE 17.1. osnova v bodech

3.4 Other criteria

We have discussed fairness as one of the criterion for evaluation and optimization of RS and ML tasks, understandably so, as it is the main topic of this thesis. But what for is an algorithm that is as fair as possible if its outputs will be disliked? We need to have an overview of other optimization criteria in order to design algorithms that will be useful in real world applications. Fairness, as well as most other parameters we are usually trying to optimise is a part of a general multi-criterion optimization which we call - the real world. Let us now introduce some of the most popular criteria that we can aim to optimize.

3.4.1 Bias

As already mentioned before, bias is an important harmful property that we are aiming to reduce. Examples of bias are all around us. Each person has at least some prejudice and false ideas about topics that they have no expertise in. It is therefore important that we actively try to broaden our minds and actively suppress these biases. Just then, we can truly become equal as individuals.

From the algorithmic view, bias is mostly introduced from the underlying data, that we are using as a training set. Sometimes these biases are even knowingly used to generate profit. For example in marketing, where sex is a very good indicator of preference. A good question arises - where lies the line between fair usage and abuse? We do not know yet, but it seems that privacy and fairness will be gaining importance and therefore the question of how to get rid of bias will become more significant.

3.4.2 Privacy

We have to extend our definition of privacy to ML systems as well. It tries to model given underlying data and therefore can leak users' information that is present in the data if the model is 'copying' the underlying data-set too well. Some models are explicitly based on similarity between users. A great example can be user-based collaborative methods from recommenders systems that were described in subsection 2.1.2. In collaborative methods we first find users that are similar to our target user to whom we are generating a recommendation and then we recommend those items that the similar users liked and our user have not yet seen. It can inherently happen that this preference we are unknowingly exploiting can be considered private. In this approach lies the great strength of user-based collaborative methods, they can extract latent meaning from data that would stay otherwise hidden. But on the other hand it can lead to a breach of privacy. Another great example can be text processing systems that are taught on a huge corpus of diverse text ranging from books, private messages, code repositories and other sources. A good example is a coding assistant for generating suggestions called GitHub Copilot. After the release of an initial version, it was discovered that it leaks secret information from the code repositories that it was trained on as discussed in [36] and [37], more specifically - API access keys. The issue was later mitigated by applying content filters that check for known secret data such as the already mentioned API access keys, emails, passwords and other personal data.

Other issues apart from private data propagation out of training data-sets to predictions of the models can be the gathering and usage of massive training data-sets it self. Fair use which is vaguely mentioned by almost all service providers that gather users data should be revised and brought up to standards of the 21st century.

Another dangerous privacy breach can be an identification of users from the published data-sets. We need to protect the gathered data and anatomize them so that the potentially private data cannot be traced back to the actual people. One major privacy leak happened with the second Netflix challenge as mentioned in [38] using methods from [39]. With only a very small amount of person's movie watch history, for example extracted from a public source such as IMDb.com, we

can match the data from Netflix challenge to this public source. The Netflix challenge was later canceled.

3.4.3 Trust

Different machine learning applications usually require very contrasting emphasis on optimizing trust. It directly corresponds with how high of an impact the provided algorithmic decision can have. We will put a system that recommends books to a way lower level of scrutiny than a system that recommends which medication or treatment should a person get. This direct proportion between the importance of machine learning output and a potential personal loss, needs to be taken into account while designing the system. Sometimes even a highly effective and correct output of a machine learning application can be dismissed only due to a lack of trust in the application. Then, the effectivity of the system can diminish even if the training and testing data tells otherwise. Trust can be greatly increased when providing explanations accompanying the output of the system.

3.4.4 Explainability

Having a reasonable explanation as to why we are getting the result we are getting can greatly improve the real-world effectiveness of the system. It can decrease negative reactions to a non-ideal predictions and make users more forgivable as stated in [40]. We provide two examples of how can such an explanation look like, first from Amazon book store 3.2 and second from Facebook's explanation of why was a particular advertisement served 3.3.

Explanations can achieve not only increased trust in the system as mentioned before, but increase in transparency, scrutability, persuasiveness, overall satisfaction and more. The main problem is, that explanations are generally hard to provide and they need to be taken into account while developing machine learning systems in all stages. Some algorithms are currently very hard or outright impossible to explain. One of the proliferated example are neural networks. Neural nets keep context in neural connections which are very hard to provide explanation for. We sometimes refer to this property of a system that we do not fully understand as "black-box" systems. We know how they work, how to train them, but due to the inner complexity we fail to explain the exact way of how the result was calculated. That leads to methods that approximate the black-box behavior and try to provide explanations with some varying degree of inaccuracy, such as [41].

Explainability can be even indispensable. Lets for assume that a ML model is used to recommend a legal action against an individual in a judiciary setting. We cannot soundly present it as an evidence without being able to justify its actions and warrant its correctness.

3.4.5 Legitimacy

Legitimacy can be viewed as an another step after explainability. When we provide a prediction together with a sound reasoning that can be verified either

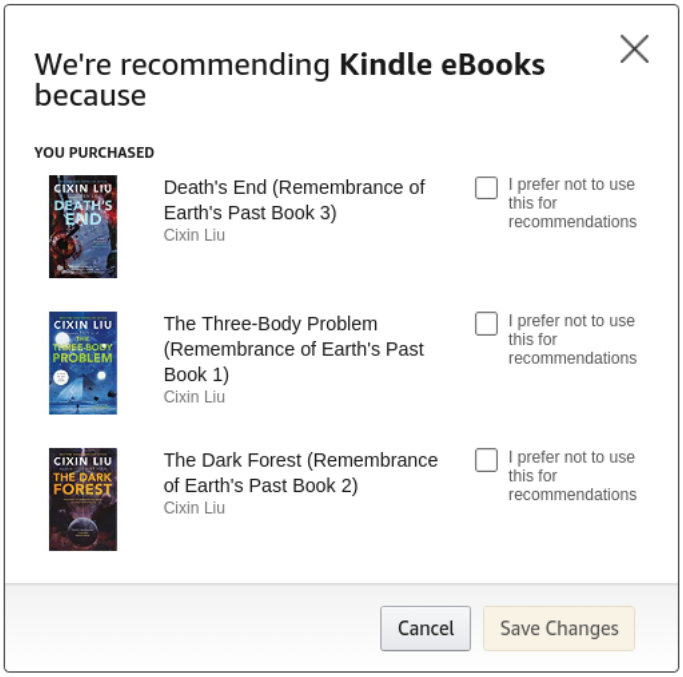

Figure 3.2: Amazon's web-page window that provides context why was a particular book recommended.

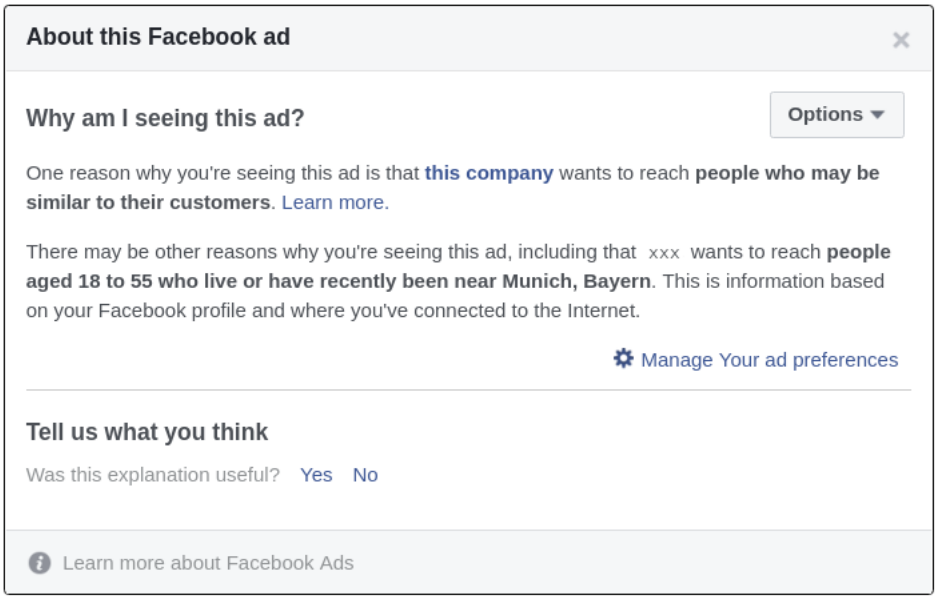

Figure 3.3: Facebook's web-page window that provides information about what data lead to user's seeing Facebook's ads.

by a human analyst or by an other trusted system, then we can use it in more serious and sensitive deployments. One of them can be the legal domain. In it, computer systems can be used to provide unbiased and fair decisions and analysis, if set up correctly. But as mentioned many times before, we need to be very careful with the data that we use for training. Lot of care needs to be taken even if we do not use an automated training solutions. Most of the current systems used in the mentioned legal domain and other systems where legitimacy is required are most probably base on hand crafted rules that can very well be subject to bias as well. One of the examples being the system COMPAS presented in 3.2.2.

3.4.6 Consistency

For some systems it is very important to stay consistent in outputs even with a slightly different inputs. Lets assume, that we are developing an illness detection algorithm, that offers medical personnel some insights about their patients based on provided symptoms. The predicted recommendation should not drastically change if for example the temperature changes by 0.1°C. It is therefore closely tied to the previous properties of legitimacy and explainability. Most high critical ML systems have to be designed with consistency in mind. Driver-less cars that abruptly change direction with only one pixel change in the input would not induce trust of its users.

From a different perspective we can view fairness as a sort consistency in the way that protected attributes of people do not have an effect on the output, the algorithm is consistent in respect of these protected attributes.

One other meaning of consistency in regards to ML systems can be stability in the meaning of ML outputs being the same while navigating a system or an app. We can attribute it to the application's design more than the ML system it self. If a user is browsing a web page, lets say an aggregator of products on the web, they may rely on the back button while browsing the different outputs. In this setting, inconsistency and changes to the recommendation items may be harmful as it will affect the user experience while listing and using the provided information.

3.4.7 Novelty

Contrary to the previously mentioned consistency parameter, we sometimes strive for novelty and exploration. If a person is picking out a movie to watch, then showing them the same selection over and over would most certainly lead to their dissatisfaction due to the limited and repeated content that the system is providing. This property is most sought for in domains where exploration is important.

Zrovna pro novelty existuje celkem dost docela zajimavych interpretaci, ktere se daji zminit. Novelty jako inverse popularity (pokud nemame zaznamy o impressions, interakcni novelty - uzivateli jsme to zatim (mockrat) nezobrazovali, temporalni novelty - je to nove vytvoreny produkt

3.4.8 Coverage

3.4.9 Diversity and similarity

3.4.10 Accuracy and precision

3.4.11 Performance

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 [17] 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.

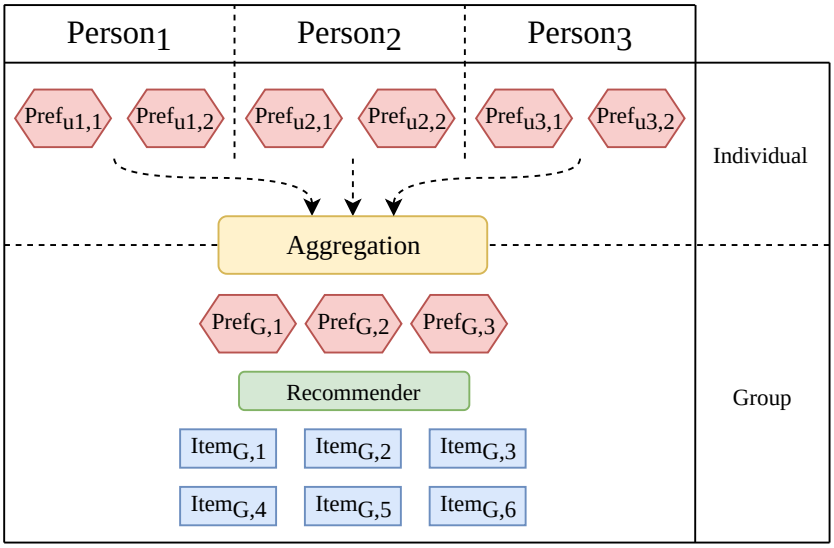

Figure 4.1: High level overview of group recommendation with aggregation of individuals' preferences, before recommendation.

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.

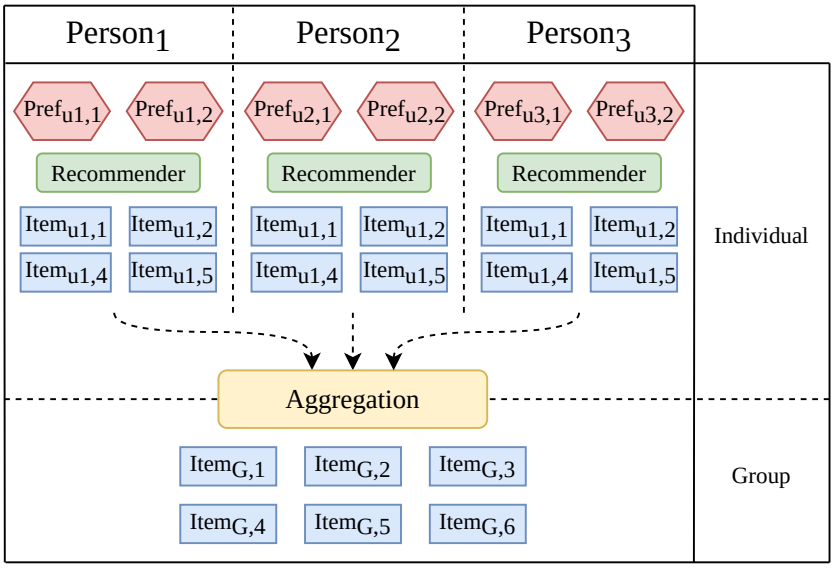

Figure 4.2: High level overview of group recommendation with aggregation on top of recommendation results for individual users.

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], [14] and [42] 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 [43]. 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[44] 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 2.1 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 [15] 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 4.17 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 4.17. 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:

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

4.3.2 Package-to-Group Recommendations

Paper [45]?

4.3.3 Top-N group RS with fairness

Paper [46]?

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

Check following:
- Introduction with clearly defined goals
- list of literature
- description of the problem that we are solving
- method how will we solve it, precise design, and science
- application of the aforementioned method
- results
- conclusion with what we found out and what are the benefits of my 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] Judith Masthoff. Group recommender systems: Combining individual models. In *Recommender systems handbook*, pages 677–702. Springer, 2011.

[15] 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.

[16] Mark O'connor, Dan Cosley, Joseph A Konstan, and John Riedl. Polylens: A recommender system for groups of users. In *ECSCW 2001*, pages 199–218. Springer, 2001.

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

[18] Fairness noun - definition.

[19] Google. Google books ngram viewer 2012 corpa.

[20] family=McAuliffe given=Katherine, family=Blake given=Peter R., and family=Warneken given=Felix. Do kids have a fundamental sense of fairness?

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

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

[23] 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.

[24] John Corbit, Katherine McAuliffe, Tara C Callaghan, Peter R Blake, and Felix Warneken. Children's collaboration induces fairness rather than generosity. *Cognition*, 168:344–356, 2017.

[25] Dana Pessach and Erez Shmueli. Algorithmic fairness. *arXiv preprint arXiv:2001.09784*, 2020.

[26] European Union Agency for Fundamental Rights, European Union. Agency for Fundamental Rights. Council of Europe, Europäische Union Agentur für Grundrechte, Europarat, Danmark. European Court of Human Rights, and Europäischer Gerichtshof für Menschenrechte. *Handbook on European Non-discrimination Law*. UTB, Stuttgart, Germany, 2018.

[27] Jeffrey Dastin. Amazon scraps secret AI recruiting tool that showed bias against women, 10 2018.

[28] Felix Richter. Women's Representation in Big Tech, 07 2021.

[29] Taylor Telford. Apple Card algorithm sparks gender bias allegations against Goldman Sachs, 11 2019.

[30] ProPublica. How We Analyzed the COMPAS Recidivism Algorithm, 02 2020.

[31] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[32] David Blaszka. Dangerous Feedback Loops in ML - Towards Data Science, 12 2019.

[33] R Kelly Garrett. Politically motivated reinforcement seeking: Reframing the selective exposure debate. *Journal of communication*, 59(4):676–699, 2009.

[34] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. The echo chamber effect on social media. *Proceedings of the National Academy of Sciences*, 118(9), 2021.

[35] Pablo Barberá, John T Jost, Jonathan Nagler, Joshua A Tucker, and Richard Bonneau. Tweeting from left to right: Is online political communication more than an echo chamber? *Psychological science*, 26(10):1531–1542, 2015.

[36] GitHub Copilot regurgitates valid secrets — Hacker News, 07 2021.

[37] Dotan Nahum. 3 Weeks into the GitHub CoPilot secrets leak - What have we learned, 08 2021.

[38] Steve Lohr. Netflix Cancels Contest Over Privacy Concerns, 03 2010.

[39] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125. IEEE, 2008.

[40] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pages 801–810. IEEE, 2007.

[41] Sainyam Galhotra, Romila Pradhan, and Babak Salimi. Explaining black-box algorithms using probabilistic contrastive counterfactuals. In *Proceedings of the 2021 International Conference on Management of Data*, pages 577–590, 2021.

[42] 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.

[43] 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.

[44] 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.

[45] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Fairness in package-to-group recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, pages 371–379, 2017.

[46] Dimitris Sacharidis. Top-n group recommendations with fairness. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, pages 1663–1670, 2019.

List of Figures

3.1 The graph shows the phrase ”fairness” occurring in the corpus of English books from 1800 to 2010. Source: Google Ngram Viewer, corpora 2012. [19] 12
3.2 Amazon’s web-page window that provides context why was a particular book recommended. 23
3.3 Facebook’s web-page window that provides information about what data lead to user’s seeing Facebook’s ads. 23

4.1 High level overview of group recommendation with aggregation of individuals’ preferences, before recommendation. 27
4.2 High level overview of group recommendation with aggregation on top of recommendation results for individual users. 28

List of Tables