Your Turn To Roll

Exploring gaps in group recommendation research

Aurél Bánsági

TU Delft

Your Turn To Roll

Exploring gaps in group recommendation research

by

Aurél Bánsági

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 27, 2021 at 1:30 PM.

Student number:	4251342	
Project duration:	October 29, 2019 – August 27, 2021	
Thesis committee:	Prof. dr. Alan Hanjalic,	TU Delft, chair
	Dr. ir. C.C.S. Liem,	TU Delft, supervisor
	Dr. Jie Yang,	TU Delft

Your Turn To Roll

Exploring gaps in group recommendation research

Aurél Bánsági

Abstract

In group recommendation, a key question is how preferences from individuals should be obtained and then aggregated into a group outcome. Collecting individual preferences can be done through implicit or explicit means, but there is insufficient research available on what option is optimal. For comparing different possible aggregation strategies, much of the existing research in the field departs from existing preference data (e.g. ratings), but considers synthetically created groups, rather than real groups. This study describes two experiments focusing on these issues. The first compared historical listening data with explicitly provided data and showed them to be similar. The second experiment compares different aggregation strategies in a more ecologically valid setting. More specifically, it considers playlist creation scenarios for groups, in which participants were asked to create their own, real-life groups. While the playlists as a whole did not show any significant differences in performance, inspecting the ratings of the tracks did show some differences. Finally, acoustic similarity was explored using the data obtained in the experiments. Some early results were observed, but more research is needed in this area.

Preface

Before you lies the thesis ”Your Turn To Roll”, which explored research gaps in the field of group recommendation by performing two separate experiments. At the start of this thesis I had minimal knowledge on group recommendation, so I truly had to start from scratch. This, combined with the start (and continuation) of the COVID-19 pandemic which forced me to re-think the set up of my experiments, made the entire process an interesting but fun journey.

First of all, I would like to thank everyone from the TU Delft that helped me during my thesis, specifically Cynthia for her supervision and guiding me through the process, Jaehun for his advice on collaborative filtering, and the team behind Nummertjes.club for providing me with a starting point on the different scoring methods.

Secondly, I would like to thank everyone that participated in the experiments, in particular the group leaders in the second experiment, some of whom had to chase some of their group members down to keep the process going. Additionally, a special thanks to everyone that helped me test the surveys, without their help the experiments would not have gone as well as they did.

Finally, I would like to thank Sofia, Daniël, Tessa, Rosalie, Andreas, my parents and sister, and everyone else who has helped me, for their support and motivation (and patience) these past two years. I could not have done it without all of you.

Aurél Bánsági
Delft, August 2021

Contents

1 Introduction

With increasingly large collections becoming available for consumption and discovery, research in recommendation is currently a hot topic. Much of the research in this area focuses on recommending items for a single user, such as a movie to watch, new music to listen to, or shopping. However, there also are situations in which it is beneficial to recommend items to a group of people. For this, so-called Group Recommender Systems can be used. Different people may have differing preferences. Therefore, for group recommenders, the key question to answer is how to aggregate the preferences of the individuals forming the group. This chapter will state a handful of problems related to this question and will then formulate three research questions.

1.1. Problem Statement

Group recommendation currently has open problems in two areas: there are questions that are yet to be answered, or might not have an answer at all, and there are some gaps in prior research. This section will discuss both sides separately. Since this research will focus on group recommendation in the music domain, problems in this area will also be stated.

Open Questions:
Before preferences can be aggregated, they first need to be obtained. This can be done through either implicit or explicit means. When an explicit method is used, people using the system are usually asked to rate a set number of items, which can be either all the available items in the system or a representative set of items. This can be a time-consuming process if the number of items that needs to be rated is high. Obtaining preferences through implicit means can be done by either "following" a user over some amount of time to learn their preferences or by using data that is already available. Some systems have also used user demographics to infer preferences. Since both approaches have benefits and downsides, it is still an open question as to which method is optimal to use.

After collecting preferences, they need to be aggregated into a group profile. Many different aggregation algorithms have been proposed for this [30], that each have their own strengths and weaknesses. However, there is no single best choice; in fact, as long as three or more items are to be aggregated, Arrow's impossibility theorem [2] states that there is no theoretically optimal aggregation strategy.

Domain-specific group recommender scenarios may also target different purposes with corresponding important criteria; for example, when recommending an expensive item (such as a location to visit for a holiday), it is important to recommend the single best item, while when recommending "less expensive" items, such as songs in a playlist, having multiple "good enough" options may suffice. Therefore, it may be interesting to consider aggregation strategies in the specific contexts of such scenarios.

Gaps in Prior Research
Often, prior research focusing on this chooses to use only a single aggregation strategy for an experiment, without the consideration of alternative aggregation options. In much of the prior work that did consider multiple aggregation strategies, these strategies were performed on data with synthetic aspects: either by approaching real users for feedback, but with synthetically created, fictional group

scenarios, or by synthetically generating groups from earlier user interaction feedback data (e.g. past consumption or ratings), while that feedback is unlikely to have occurred in the context of group scenarios. Only considering such synthetic situations can be risky as, according to [30], synthetic data can lead to inaccurate results if the wrong metric is used. The domain of a group recommender system also greatly influences these results, meaning research in one domain is only sometimes applicable in other domains.

Problems Specific to the Music Domain
More specific to the music domain is the way group recommender systems acquire music preferences of group members, the most common method is to use the ratings of individual items. These individual ratings can be retrieved through either explicit or implicit methods and, when collected properly, are representative of the music taste of a person. However, this method assumes two things: independence between the items and that the taste of a person alone matches with that of the same person in a group. The latter is fairly straightforward, but the former is more nuanced. When a person selects a number of items (or a number of items is implicitly obtained), it is very likely that those items are related to each other in some way. Recent research has shown that people unintentionally use audio features to curate a playlist, preferring tracks that have acoustically similar features [11]. While this research was done for playlist creation and recommendation on an individual basis, it is likely that the results also hold to some degree in group recommendation.

1.2. Research Questions

This thesis will consist of two experiments answering three research questions. The first experiment will focus on the collecting preferences questions and will answer **RQ1)**. The second experiment is the main focus of this thesis and will compare different aggregation strategies in **RQ2)**. The data from the second experiment will also be used to research acoustic similarity in **RQ3)**.

- **RQ1)**: How can historical data be used to improve group recommender systems in the music domain?
- **RQ2)**: To what extent does the performance of a group recommender system vary when considering different aggregation strategies?
- **RQ3)**: To what extent can the acoustic similarity of playlists, recommended for a group, be measured and increased?

Chapter 2 dives into the literature of group recommenders to give more context to the research questions. **RQ1)** will be answered with data obtained from the first experiment in chapter 3. **RQ2)** and **RQ3)** both require data obtained a group recommender in a real-life setting. The system design and set up of the group recommender will be discussed in chapter 4. The two research questions, **RQ2)** and **RQ3)**, are discussed in chapters 5 and 6 respectively. Finally, the results of the experiments and a reflection of the results are given in chapter 7.

2 Literature Study

This chapter will discuss how group recommenders work and the different aspects of group recommenders. Because being in a group naturally causes group dynamics and other psychological effects to occur, these will also be discussed. This chapter will also discuss a selection of related work and different streaming services that can be used for the experiments of this thesis.

2.1. Recommenders

Since group recommender algorithms usually build on top of single-person recommender algorithms, it is important to know how those operate and what different types of algorithms there are.

Early research in recommender algorithms was done in the early 90s, with works such as Tapestry [18] and GroupLens [50], the latter describing a system that recommends interesting articles for a user in newsgroups. By the end of the twentieth century, the field had grown enough to have papers comparing different systems, such as an article written by Resnick et al. [49].

A few years later, in 2006, one of the key players in recommender systems, Netflix, held a contest in which teams were challenged to beat their algorithm [6]. Only after three years did a team beat the challenge, but this did create extra interest in recommendation. In current times, recommender systems are seen as important and widely used. In fact, platforms, such as Netflix or YouTube, have even build their business model around it, as better recommendations result in higher participation.

Two main categories have emerged from research: collaborative filtering, which focuses on the user, and content-based filtering, which focuses on the items. Since the focus is different, their general approach is also different. It is also possible to combine different algorithms of the same or even of two different types, such as in [44] or the winner of the Netflix prize [57]. Since hybrid systems can use the strengths of both categories and often mitigate the weaknesses, they often outperform non-hybrid systems.

2.1.1. Collaborative Filtering

Collaborative filtering relies on an assumption that people who agreed in the past will continue to agree in the future. For example, if two people both like similar music and one of them likes a new song, the other person will have a high chance of liking that song too. Collaborative filtering generally works by taking user ratings, implicit or explicit, and finding other users with similar taste. Schafer et al. [53] offers a good overview of collaborative filtering. Collaborative filtering is only applicable when the dataset matches certain criteria. For example, it is important for the dataset to have many items and have more users than items. Tapestry and GroupLens are both examples of early collaborative filtering systems.

An issue with collaborative filtering is scalability. As the number of items and users grow, the complexity of the algorithm grows too. For example, the “Who to follow” system at Twitter, which recommends potential users to follow, was designed to work on a single server in 2013 [20]. While at the time of writing the memory requirements were possible, a server with 144 GB of RAM was used, the authors quickly realised that they would run into scaling issues. The common way around this, which the authors also used, is to use a server cluster to distribute the task at hand. Of course, this is more difficult to set up and also comes with a higher financial cost.

A real world example of a recommender algorithm directly using collaborative filtering is Spotify's Discover Weekly playlists. Each Spotify user will get a new playlist of songs they probably do not know, but that they probably will like. The algorithm that creates these playlists looks for songs that a user has rated highly in the playlists of other people that have a similar taste profile to yours. In these playlists, songs that are unknown to the user but do fit their profile are added to the playlist. Using this approach, a user gets presented new music that should be recognisable and enjoyable [41].

2.1.2. Content-based Filtering

While collaborative filtering needs a lot of information on the taste profiles of users, content-based filtering works on the opposite: it assumes there is a lot of information on the items, but not much on the users. Lops et al. [27] provide a high-level overview of content-based filtering algorithms. Content-based filtering works by creating an item profile for each item, which can then be retrieved either by using a user profile, generated in a similar fashion, or by user query. These features need to be abstracted in order to make them easier to work with, this is commonly done with algorithms like *tf-idf* [51]. Finally, the recommender uses this information to recommend similar items to what the user has enjoyed in the past.

The main issue with content-based filtering is that algorithms are often content-specific. For example, a system that first captures the taste of a user based on their music preference and then recommends new music is useful, but not as useful as a system that can then also recommend videos, news, or other different content types.

Since collaborative filtering has become more popular, there are no more big examples of systems just using content-based filtering. However, as will be discussed next, content-based filtering is very powerful in mitigating issues with collaborative filtering. Due to this reason, content-based filtering is now most often used in hybrid systems in combination with other algorithms.

2.1.3. The Cold Start Problem

The biggest issue with recommender systems is the cold start problem. This issue can occur in three different situations when the systems needs to return a recommendation: for a new user, on a new item, or for a new user on a new item. For example, if a new user joins and wants to get a recommendation, the system is not yet aware of their preferences. A common solution is to predict the taste of a new user through some method, for example in [26], which puts new users into groups based on their demographic while the system learns about their preferences.

Similarly, a new item will initially not get recommended since no users have rated this item yet. This might also occur for items that are not new, but also have no received sufficient ratings. This means an item can get recommended, but the recommendation is likely of low quality, which causes less visibility. As content-based filtering is based on the properties of items, and thus not on the popularity of an item, this is mostly problematic for collaborative filtering. This further shows the strengths of developing a hybrid system.

2.2. Group Recommenders

Adding in more people to a recommendation obviously only complicates the process of generating recommendations. Rather than having to find the preference of a single user and using this to recommend items, it is now required to find the preferences of multiple users and use these to find items that are enjoyable to the group as a whole. After a group recommender has recommended one or more items, it might be important for the group to discuss and chose a result, rather than being given an item directly. This section will show how group recommenders work, how they differ from individual recommenders, and how psychological effects can change the results.

2.2.1. Group Recommender Dimensions

Like with individual recommenders, there are different categories of group recommender systems. In fact, Masthoff [30] found that there are six different dimensions, which can be used to categorise a group recommender:

- Individual preferences are **known** versus **learned over time**

- Recommended items are **selected by the system** versus **presented as a choice**
- **Passive** versus **active** groups
- The recommendation is **a single item** versus **a sequence of items**
- The method of estimating ratings, usually **collaborative filtering** versus **content-based filtering**
- The method of rating aggregation

Group recommenders have an active group when they allow users to interact with rating aggregation, for example by letting a group make the preferences of a specific group member more important. There are also other possible dimensions, such as whether groups members are present in the same place, or how stable a group is with respect to the group members. These are however less discussed in research. The method of rating aggregation is elaborated further by Jameson and Smyth [23], which states that there are generally three distinct ways to aggregate ratings. Each option represents a different point in the process where aggregation happens.

The first method is to recommend items separately for each user and then to combine these recommendations by use of some aggregation method. This approach is easy to build on top of a pre-existing individual recommender, but it is unlikely that the system returns a recommendation that is optimal for the group as a whole. Interesting point is that while Jameson and Smyth state that this approach is rarely used and sub-optimal compared to the other two approaches, it is still commonly mentioned by more recent work.

The second option of rating aggregation is to aggregate the *ratings* of each user. This effectively creates a new pseudo-user that resembles the taste of the group as a whole. Since this transforms group recommendation into recommendation for a single person, existing techniques for individual recommendation can be used. As with the first option, this approach still requires individual ratings to be estimated through some way. Examples of systems using this option are described in [12, 33, 38].

The last method is aggregating even earlier by creating a group model. This is a model of the preferences of the group as a whole, which means that no individual ratings have to be estimated anymore. Creating such a model is a tough task, since it is very domain specific and time consuming to get right. Benefits are that using a group model, when done well, offers better results and explainability and also has more privacy than other options. INTRIGUE [1] and Hootle [28] are examples of systems constructing a group model for tourism, which appears to be the most common domain in which this option is used. An example of a system using group modelling outside of the tourism domain is the GAIN system [14], which focuses on showing suitable news and advertisements in fitness centres.

Since this thesis will later focus on the music domain and passive groups, group modelling is less viable. For this reason, the second option, aggregating ratings, will be discussed further and used for the group recommendation experiment.

2.2.2. The Steps of a Group Recommender Algorithm

Regardless of the different dimensions of a group recommender, they generally have the same steps a system must undertake to reach a suitable recommendation. Jameson and Smyth [23] found that most group recommenders have the following four steps:

1. Acquiring individual preferences
2. Generating recommendations
3. Presenting recommendations
4. Helping the group reach consensus

It should be noted that not each system will have all four steps. The last step in particular is optional, since not all group recommenders offer a choice to the group, they just offer an item (or sequence of items) with no other options. Nevertheless, each step will be described separately.

Step 1: Acquiring Preferences:
Collecting the taste of a person can be done either explicitly or implicitly. When asking explicitly, the users are asked to either list some items they like or to rate a number of items given by the system. In MusicFX [33], members of the fitness facility were able to rate all radio stations, which were used directly by the system. Similarly, in PocketRestaurantFinder [32], users are asked to rate features, such as cuisine or cost, of different restaurants. Collecting preferences explicitly is very powerful since it enables the system to directly learn the taste of people. However, a downside is that it requires effort from participants, which is also often time consuming.

Learning taste implicitly can be done through different ways. Flytrap [12] learns the preferences of a person by storing what songs they are listening to on their own. The stored tracks and artists are then used to award points for tracks the person has not listened to yet. It is also possible to acquire tastes even more implicit, for example by using demographics to divide users into groups automatically. INTRIGUE [1] does this by classifying each person as belonging to a subgroup. For each subgroup, the suitability of a location is calculated using numerous factors.

In domains where the number of available items is too big to have a user rate each item (either implicitly or explicitly), techniques from individual recommender algorithms can be used to estimate ratings. [39] and [4] use collaborative filtering to estimate ratings for items unknown to the user, but of course, content-based filtering can also be used if it is better suitable for the domain.

The relevant dimension for this step is if individual preferences are known beforehand or developed over time. An example of the latter is Adaptive Radio [9], which is a radio station that continuously plays music. Initially, the system has no taste profile of a user, but as they listen to the radio, it learns about their preferences. Interestingly, the writers chose to use negative feedback for this, in the sense that users cannot “like” a song, but only vote to skip it. The writers argue that by creating a filter for each person, the tracks that remain will be, at least somewhat, enjoyed by everyone. This is similar to the *Least Misery* aggregation strategy, which will be discussed later. In offices, which the researchers focused on, the resulting music was not distracting, which was the goal of the paper, but was also described as bland.

Step 2: Generating Recommendations:
After the preferences have been collected and estimated, they need to be aggregated to a single rating per item for the entire group. This group rating then allows for the actual group recommendation to take place. Which method of aggregating to use is perhaps the biggest question in group recommendation currently. Since there are so many different aggregation strategies, they will be discussed separately in section 2.4. If the group should be able to interact with the rating aggregation (i.e. the group is **active**), the system needs to provide some method of interacting during this step.

Aside from aggregating ratings, there are other important concepts in this step. Often, it is important to be able to explain how the system got to a specific recommendation. While explaining the recommendations itself is part of step three, it is impossible to offer a good explanation if the algorithm was not designed with explainability in mind. In algorithms where a deterministic aggregation strategy was used, this can be quite simple, as long as the strategy is understandable for humans. There are, however, also systems that use a non-deterministic design, which changes strategies based on conditions. Senot et al. [55] build a system that compares aggregated ratings with reference group profiles and selects a strategy based on that. In [10], a genetic algorithm was used to predict group interactions and ratings. In both of these cases it would be difficult to explain why a choice was made, and neither paper mentions explainability in their experiments.

Another consideration is whether the group recommender should recommend a single item or a sequence of items and if the resulting items are a choice or the only option. Both of these dimensions are very domain dependent. In the tourism domain, most systems [1, 22, 37] present a number of individual items as a choice. In the music domain, it is less common to offer an option, but the system can recommend either a single item such as a radio station in MusicFX [33] or a sequence of items, for example in Flytrap [12].

Step 3: Presenting Recommendations:
The third step is the first optional step. In this step, the system presents and explains the recommendations it generated. In systems that present only one option, this step can be to just display the item, for example by starting a recommended track on a music player.

Nevertheless, explaining recommendations to the group as a whole, or to the individuals, will usually improve the performance of the recommendation. If the system was designed correctly in the previous step, it should be possible to point out relations between selected items and specific group members (e.g. “This item was chosen because Bob really likes this”). An explanation can have different criteria [47] and different types [36]. Two of these types, “reassuring” and “repairing inconsistencies” try to influence the emotions of a person, for example by letting them know that this item is rated high by another person in the group. More information about the effect of emotions in group recommendation can be found in section 2.5. The last category, preserving privacy, is also very important. A trade off needs to be made between allowing transparency by acting on the emotions of a person and exposing the ratings or items from the group members. This is especially important, as a lack of privacy can influence the first step, collecting preferences. If the system exposes too much information, it might lead to negative thoughts or people reading into the recommendations too much [47].

Step 4: Helping a Group Reach Consensus:
The last step, helping the group reach consensus, is rarely used. In most cases, the recommender system will either recommend a single item (in which case the previous step was potentially skipped too) or assume a single person will make the final decision, such as in INTRIGUE [1]. Only in some cases does a recommender system help with consensus reaching. STSGroup [37] has chat functionality in which group members can suggest and discuss different options. Another example is the Travel Decision Forum [22], in which group members can chat between themselves and agents which represent absent members.

There is also research that studied how people discuss amongst themselves when having to chose an item. In a study by Delic et al. [16], students were asked to pick a location to visit. The results of the experiments showed that the groups often picked another location than the one predicted by the group recommender built for the experiment. The results also showed that most people were happy with the chosen location, even if it was not their own choice. In an experiment by Masthoff [30], participants stated that their choices were influenced by wanting to prevent misery and starvation (i.e. making sure that everyone had a say in the recommendations).

2.3. Music-specific (Group) Recommendation

The music domain is quite unique for group recommendation and this causes some factors to have a different priority than most literature suggests. In most domains, it is important to find the best item for a group, for instance what city to visit, as the cost of a wrong recommendation is quite high. However, in the music domain, it is less important to find the best item, as this cost is substantially lower. Rather, it is important to include serendipity in the recommendations and to provide a sequence of music that makes sense. Since the cost of items is lower, it is also not needed to help the group reach a consensus. The third step of a group recommender, explaining the recommendations, is also not strictly needed, but can still be used to improve the performance.

McCarthy and Anagnost [33] show this principle with MusicFX, a system where people are able to rate different radio stations in a fitness centre. The system then sums all ratings together for each person present and uses these scores as weights for a randomised algorithm. Adding randomisation prevents the recommender from picking the same item every time when the same people are present.

Similarly, Flytrap [12] also normalised ratings into a probability density function. As mentioned before, tracks are given a high score if the user has listened to it before or if it fits their preferred genres. They also created an automatic DJ that can alter the probability to make sure the created playlist still had a good taste. For example, the chance of getting two tracks by the same artist in a row is very small. This strategy of using ratings as weights in a probability density function was more thoroughly examined in a paper written by Popescu [42]. In this paper, he calls this strategy *Probability Weighted Sum* and argues that it is easy to compute and encourages truthful answers from people. In a small scale experiment comparing different strategies, it was found that this strategy showed a tiny gain in ratings.

Recommending Sequences of Items
Group recommenders in the music domain will also commonly recommend sequences of items (i.e. playlists) over single items. When recommending sequences, the performance of a system will depend on more factors, as the recommended items will be perceived to have a relation between each other.

Specifically, a sequence should have: a good narrative flow, mood consistency, and a strong ending [13, 30]. In the paper, these concepts were explained using a news TV program, but they can be easily adapted to music. Narrative flow and mood consistency are highly related, especially in the music domain. Narrative flow is leaning more towards recommending tracks of the same artist or album close to each other, while mood consistency means that similar sounding tracks should be close. Ending a sequence with a highly rated item also improves the performance as the ending of a list is the most memorable. Creating a strong ending comes with its own challenges, which will be briefly explored in section 2.5.

Finally, there are also aspects of recommendation in the music domain that are not specific to group recommendation. Music can be a strong influence on the emotions or mood of a person listening, for example by making someone feel less alone or by putting someone in a good mood [54]. This influence works both ways, which means that there are many factors, such as personality, mood, that change how a person enjoys a song. Other factors that could be relevant are the current activity and social environment of a person. Recent research has shown that there is a link between audio features given by Spotify and different personality traits [34]. This link between the context of a person and recommending music is starting to be explored, for example in [19], which re-ranks recommendations based on the time of listening.

2.4. Aggregation Strategies

One of the key questions to answer, when making a group recommender that uses rating aggregation, is what strategy to apply when aggregating these ratings. Since this problem is similar to that of a voting problem, Arrow's impossibility theorem [2] states that there is no optimal algorithm to use. Research thus often focuses on new aggregation strategies or ways to improve existing ones. Masthoff et al. [29] lists commonly used aggregation strategies and their strengths and weaknesses. This paper also states that the *Multiplicative*, *Average*, *Average without Misery*, *Most Pleasure*, and *Borda Count* strategies often perform well.

There is an interesting phenomenon that happens when research uses synthetic data to compare different strategies. These experiments, for example papers by Recio-Garcia et al. [48] or Senot et al. [55], often find the *Average* strategy to be the best. Masthoff notes that this could be due to these experiments calculate the group satisfaction by averaging the satisfaction of the individual members. This leads to the *Average* strategy outperforming the others, which would contradict Arrow's impossibility theorem. The benefit of using synthetic data is that it is easier to compare different strategies, which experiments using real data often lack. In fact, most research with real-life experiments focus on one strategy and do not consider any others, for example in MusicFX [33] or PolyLens [39].

When people are asked to reach a decision as a group (i.e. without the use of an algorithm or system), the most commonly used strategies are *Average*, *Average without Misery*, or *Least Misery* [30]. As mentioned earlier, participants stated that they care the most about preventing misery and being fair, which falls in line with the strategies that they used.

There are too many aggregation strategies to discuss or compare in this document, especially since any voting system could potentially be used to aggregate ratings. Due to this reason, the rest of this section will focus on three strategies in particular: the aforementioned *Probability Weighted Sum*, *Fairness*, and *Least Misery*. *Fairness* and *Least Misery* are both more simple to understand, but they also represent important qualities of a group recommender.

2.4.1. Probability Weighted Sum

This strategy was first used by MusicFX [33] as an extension to the *Average Without Misery* strategy. The authors used the resulting scores of the algorithm as weights for a probability density function. This approach is most commonly used by group recommenders in the music domain. Later, Popescu [42] formalised this approach and called it *Probability Weighted Sum*.

The algorithm starts by summing the squares of each rating per track, this is to make the extreme values stand out more. The squared ratings are normalised by dividing each value by the total sum of all ratings. These normalised values are then used as weights for a probability density function, from which tracks are then randomly pulled from.

This process has the effect that tracks with a higher score are more likely to be picked, but there is no guarantee. Similarly, tracks with a lower score still have a small chance to be picked. This prevents people from being able to lock out certain tracks completely by giving them a very low score. Additionally, it can increase the serendipity of the resulting playlist.

Different variants
The principles of *Probability Weighted Sum* are used in many group recommendation systems, which also means there are different variants. Equation 2.1 shows how MusicFX calculates ratings. Each radio station, or category, has a rating between -2 and 2 for each person. A value of 2 is first added to each rating changing the scale to be between 0 and 4. The ratings are then squared before being summed together. Squaring the ratings increases the distance between well-rated and less well-rated stations. The resulting scores that are the result of the equation are then sorted in descending order. The system will then randomly pick a radio station from the top rated stations, using the scores as weights.

Equation 2.2 shows the equation used by [42], the paper that also coined the term *Probability Weighted Sum*. The main difference in this approach is that the ratings are not squared before being summed. It also formalised calculating the probabilities by explicitly dividing the combined score for a track by the total score of all tracks.

Both equations are slightly modified versions of the equations as they appeared in the respective papers. The only modifications were made to ensure that the variables matched between the equations and those later in this document.

$$GP_i = \sum_{u \in U} (\text{score}(u_k, t_i) + 2)^2, \tag{2.1}$$

$$p(t_i) = \frac{\sum_{u_k \in U} \text{score}(u_k, t_i)}{\sum_{t_j \in T} \sum_{u_k \in U} \text{score}(u_k, t_j)} \tag{2.2}$$

where: t_i, t_j = A track
u = A user
T = The set of tracks
U = The set of users
GP_i = Overall group preference of a station

2.4.2. Fairness
While *Probability Weighted Sum* is a non-deterministic algorithm, *Fairness* is deterministic. Masthoff [30] describes this strategy, which starts by selecting a random user and picking their top rated item. The algorithm then picks the next user in the session and picks their favourite item. This continues until a certain number of, or all, items have been selected.

This process ensures that each person gets the same number of items in a recommendation, which is something that Masthoff [30] found is important to participants. However, the algorithm does not use the tastes of the other users, so some items might cause more misery than intended. There is also less chance of being able to “build a narrative” in the selected items, as each successive item must come from a different person.

A more extreme version of this algorithm also exists. In this variant, the next person selected is not the next one in the list, but rather the person who had the lowest rating for the previously selected item. In this case, there can potentially be less misery overall, however this version is not documented or used much.

2.4.3. Least Misery
Least Misery [30] is the easiest of the three algorithms. As the name implies, it minimises the dissatisfaction of all members of the group by maximising the lowest ratings. The algorithm works by first selecting the lowest rating per user per item. This resulting list is then sorted in a descending order. This means that the “best of the worst” item is ranked high and thus that there is a high chance that

all users will at least somewhat enjoy that item. A common criticism of this approach is that the result lacks daring choices or outliers, which means the result might be considered enjoyable, but boring. It resembles picking a popular radio station or a top 40 playlist.

2.5. Psychological Effects

Another key difference that makes group recommenders more complicated than individual recommenders is that the presence of other people adds group dynamics and other psychological effects. These group dynamics can change the behaviour or preferences of people. In some cases, there might even be an incentive to manipulate the system, so that a person can get priority on their tastes. This can mean that experiments performed with real people will always have, at least somewhat, different results compared to simulated experiments. The other side of the coin is that it is also possible to use the personality of people in different ways to improve the performance of group recommender algorithms.

Emotional Contagion and Conformity

The strongest effect is perhaps *emotional contagion*. Emotional contagion is defined as “the tendency to automatically mimic and synchronise expressions, vocalisations, postures, and movements with those of another person’s and, consequently, to converge emotionally” in [21]. In other words, people tend to have their mood and taste influenced by that of another person, especially when those people have a close relationship. Masthoff [29] notes that the strength of this effect depends on the type of relationship with the person. Out of the four basic types of relationships [17], emotional contagion is strongest when the relationship you have with the other person is of the communal sharing or authority ranking type.

Another effect is *conformity*, which can cause a person to change their opinions to better fit in with a group. There are two types of conformity: normative influence, where people only express a different opinion, and informational influence, which actually changes the belief of a person. In a famous experiment, Asch [3] tested this by asking participants to select the line that was the same length as a given line. This was done in a group, but all but one person were part of the experiment team. This experiment showed that a large minority selected the wrong answer to fit in better with the group. Conformity can both occur in the first step (acquiring tastes) and the fourth step (helping the group reach consensus) of a group recommender. In both steps, group members might not rate the items they actually like high in fear of the group disliking these items.

Individual Effects

There are also effects not related to a group setting that can still change the performance of a group recommender. Masthoff and Gatt [31] describe that the mood of a person can change their judgement, in both a positive or negative sense. The first item of a sequence of recommended items can change the judgement of the entire sequence.

Another factor that can influence emotion is expectation. When people are given a positive expectation for something (e.g. that a joke will be funny), they will also give a higher rating than if they did not have that expectation. This effect is called assimilation, while the opposite (a negative expectation lowering the rating) is called contrast. The authors state that it could be possible that by creating a positive expectation with recommending several liked items in a row, assimilation could cause the next item to also be rated higher, even if it should be liked less. The last effect they mentioned is that happiness wears off over time, even if the recommended items are well liked, which they call emotional evanescence. A series of positive items can set a new standard, which people use to compare new items to. These new items will then be rated more strict as the bar has been raised, which can then cause a negative experience.

Manipulation is also an issue that should be accounted for. For example, in MusicFX [33], people could input their own ratings for different music genres. In the first version of this system, there was a threshold for ratings, meaning that genres below a certain score would never be selected. After people noticed this, some abused this by ranking all genres, but the ones they liked, below this score. This resulted in the system only selecting the genres they wanted to listen to. While manipulation occurs more often when the system uses explicit input, it is still possible with implicit preference inference. A possible solution is to ensure that a user is not incentivised to provide false ratings. This can be done by for example by using the median of the all group members. In that case, if someone attempts to lower the rating of an item the rest of the group likes, it will have no effect to the recommendation. Another solution is to make it tough (or impossible) to manipulate the ratings, for example by using implicit preferences.

Using Personality Tests
More recently, personality has been used to strengthen the performance of group recommenders. Research currently focuses on detecting personality traits and having that aid the decision making. A test that has been in recent use is the Thomas-Kilmann Conflict Mode Instrument [24]. This test describes the behaviour of people in two dimensions, assertiveness and cooperativeness. These scores of these dimensions are then used to classify the personality of a person in five categories: competing, collaborating, compromising, avoiding, and accommodating. These personality traits are used in some recent work, such as [45, 48], to propose new aggregation strategies. The two dimensions are combined into weights, which are then used in the aggregation strategies to avoid conflict between group members. One of these is a variation of the *Least Misery* strategy, which uses the weights to change the estimated ratings of items. Since it is more likely that assertive people will experience misery when their items are not recommended, the system will lower the estimated ratings of items coming from non-assertive people. This idea was further developed in [46], which also takes into account different factors calculated from social media. The same paper also experimented with taking past recommendations into account when calculating a new recommendation. Nguyen et al. [38] used the same test to compare the performances of different aggregation strategies when groups had either matching or different personality types.

2.6. Related Work

In order to show how theory is applied in real-life systems, this section will describe some group recommender systems.

2.6.1. MusicFX

MusicFX [33] is a group recommender in the music domain and was made for a fitness centre that is only open for residents. People can input or update their preferences for 91 genres of music on a five point Likert scale. The system keeps track of who is present by people swiping a badge or signing in on a computer. These sign in events, together with others, are also used as triggers for the algorithm to recommend a new radio station. As described in section 2.3, the system converts the ratings into weights for each station. This makes the radio stations that signed-in people like the most more likely, but not guaranteed, for variety.

The results were that the majority of the participants liked the music selection better than before the experiment. Common complaints were that the system changed stations abruptly (e.g. in the middle of a song), and that sometimes songs with offensive lyrics were played. In a more quantitative evaluation, log files were used to evaluate average ratings per person per session. This was calculated by looking at the music that was played during workout sessions for each person, and averaging the ratings for these stations. This analysis showed that most people would have had a positive experience, with the only outliers rating almost all radio stations with the lowest scores.

As mentioned in section 2.5, MusicFX initially had troubles with manipulation. As there was a threshold active that would change the music if anyone rated the current playing station with the lowest rating. Some users abused this by rating the radio station that was currently playing with this lowest score, causing the system to immediately playing something else.

2.6.2. Flytrap

Another group recommender in the music domain is Flytrap [12], an online environment where the system automatically plays music based on the people who are logged. Unlike MusicFX, this system collects preferences implicitly, by tracking the tracks and artists that users are listening to, which is then used to score new tracks based on matching artists or genres. The algorithm represents the users by agents, who vote for items in their stead. Similar to MusicFX, these votes are then combined and used as weights in a probability density function. Finally, there is an extra agent acting as a DJ, making sure that the recommended items make sense, for example by making sure that the same artist is not played twice in a row or that there are no sudden genre switches. Basing the recommendations on genres make it easy to explain the recommendations, as the system can simply say that the track belongs to a genre a user enjoys.

2.6.3. INTRIGUE

INTRIGUE [1] is an *active* group recommender, which recommends potential places of interest to a tour guide for their group. It is also one of the first papers to use group modelling, rather than aggregating preferences or recommendations. The group model is based on dividing the group into different subgroups, such as children or people with an impairment. The system uses reference data for the subgroups to recommend either a sequence of places for each subgroup or for the group as a whole. As the system uses properties for the places it can recommend, it can also use these to generate explanations. The authors write that at the time of writing, the explanations focused on highlighting the properties that were positive for each subgroup. They also wanted to experiment with showing the negative properties, so that a tour guide can anticipate on potential problems.

2.6.4. STSGroup

Another group recommender made for tourism is STSGroup [37]. This system allows groups to suggest, and discuss different places to visit in a group chat. Users can also specify other relevant variables such as their mood. As the group chats, the system uses these logs to create a taste profile for each user. Rather than basing this taste profile on the specific places that were suggested, it uses keywords associated with these places. This lets the system predict a general taste profile quickly. The algorithm uses the *Average* strategy to aggregate the ratings, but augments this with weights for each user. These weights are based on several things, such as active users or vulnerable users being assigned a higher weight.

2.6.5. TV4M

TV4M, also called Yu's TV Recommender, is a system that recommends TV programs based on the people that are currently watching. This system uses rating aggregation to generate items, with the ratings being estimated both implicitly and explicitly. The first is done by observing the people currently watching the programs and detecting how they react. Additionally, the algorithm also assigns weights to different features for each person, which are based on how important that feature seems to be for the person. The authors performed multiple experiments which showed the effectiveness of the program and that it outperforms a random recommendation. Interestingly, they also compared it against a system that aggregates recommendations, which showed that this is inferior to aggregating ratings.

2.6.6. The GAIN System

The GAIN System [14] uses group modelling to recommend news and advertisements to people in a fitness center. It takes two groups of people into account, those for who the system knows is present and those who are statistically likely to be present. The people for which the systems knows that they are present can update their preferences on a mobile version of the program. These preferences, together with demographics data to represent the unknown group, are then used in the group model. This group model then recommends different news items or advertisements in the fitness center. Similar to MusicFX, the system does a new recommendation when a new person enters or after some time has passed.

2.7. Streaming Services

In order for the experiments to function, a streaming service needs to be used. This streaming service needs to have an API that provides historical data, allow for search, and offer some way to compare tracks, likely through providing audio feature data. The service should be popular in Europe, as that is where most, if not all, people participating in the experiment will be located. Having a popular streaming service will allow more people to participate. Finally, the service providing a recommendation API is an optional benefit, as this allows for more data to be collected in the experiments.

2.7.1. Comparing Streaming Services

The first step in finding a suitable streaming service is to look at their popularity. According to research by GlobalWebIndex [5] and MIDiA [35] the most popular streaming services in 2018 were: Spotify,

Table 2.1: Comparison of different streaming services.

	Free?	**Audio Data**	**Historical Data**	**Search**	**Recommender**
Spotify	Yes	Yes	Yes: tracks & artists, three time ranges	Yes	Yes
Apple Music	No	No	Yes: recently played, “heavy rotation content”	Yes	Yes (no seeds)
SoundCloud	Yes	No	Yes: recently played	Yes	No
Deezer	Yes	No (full tracks available)	Yes: recently played	Yes	Yes (no seeds)

Apple Music, Amazon, SoundCloud, and Deezer. Specifically Spotify, SoundCloud, and Deezer were the most popular in Europe. A comparison between these streaming services can be found in table 2.1. Amazon does not provide an API for their streaming service and will be excluded from the comparison. The other four options all have a public API and a search function for tracks, artists, and albums. Each streaming service provides historical data, but only in the case of Spotify is it possible to go back further in time. Three different time ranges are provided: short (up to four weeks), medium (up to 6 months), and long (several years). Both tracks and artists can be requested using this function. SoundCloud, Deezer, and Apple Music provide “recently played” data and the latter also provides *heavy rotation content*[1], but does not give an explanation of how this data is collected.

Apple Music and SoundCloud do not provide audio feature data or other ways to compare tracks by use of audio feature data. On the contrary, Spotify provides a function that provides multiple values per given track. Notably, Deezer allows for full tracks to be downloaded, as long as the logged in user has a “premium” account. This could potentially be exploited to compare tracks with the actual audio data, for which many different methods are available. Both Apple Music and Deezer have a recommendation API, which recommends tracks based on a user profile. It is not possible to provide seed items to the recommender system, which means that it is also not possible to make an recommendation that is not based on personal data. Spotify uses a different approach and seems to not use user data at all. Rather, seed items are required to make a recommendation. Finally, SoundCloud does not provide a recommendation API.

Due to the limited API functionality, Apple Music and SoundCloud are not suitable options. Deezer is possibly an option, especially due to full tracks being available, which allows for more direct comparison methods than audio feature data. However, given that this option is only available for paying users, the other API functionality that Spotify provides, and the larger popularity of Spotify, the latter is the best choice for user in both experiments.

2.7.2. Audio Features

Having selected Spotify as streaming service for the experiments, the next step is to study the audio features the API provides. As there are no large datasets with audio data of popular songs, this is especially important, as it is the only viable way to compare tracks. One commonly used option is to calculate features that describe the song. As mentioned before, Spotify offers an API to retrieve such features[2], which returns values representing concrete information such as the tempo or key, but also more abstract data, such as how acoustic a song is. The more abstract data, eight in total, are the most interesting, since they likely contain information on why a track is enjoyable for some. Below, a brief summary of each of these eight features is given:

- **Acousticness**: Represents how acoustic a track is
- **Danceability**: Represents how suitable a track is to dance to
- **Energy**: Measure of intensity and activity
- **Instrumentalness**: Predicts if a track is instrumental
- **Liveness**: Detects presence of an audience in the track

[1] `https://developer.apple.com/documentation/applemusicapi/get_heavy_rotation_content`
[2] `https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-audio-features`

- **Loudness**: Overall loudness of a track in decibels
- **Speechiness**: Detects whether a track contains spoken word
- **Valence**: Represents music positiveness of a track

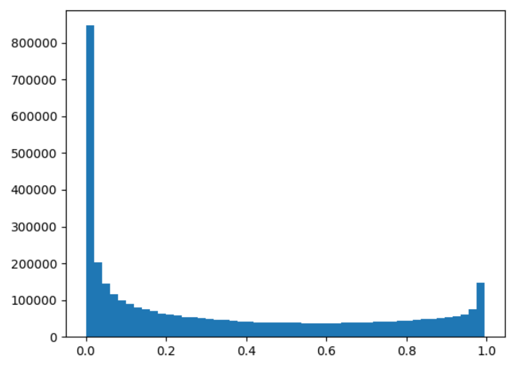
These features all have values between 0.0 and 1.0, except for *loudness*, which has a range between -60 and 0 dB. Since the audio features will, at a later stage, be used for comparing tracks, it is important that each audio feature has the same range. This can be achieved by scaling the *loudness* feature to the same range using equation 2.3.

$$\text{scaled\_loudness} = \frac{\text{loudness} + 60}{60} \tag{2.3}$$

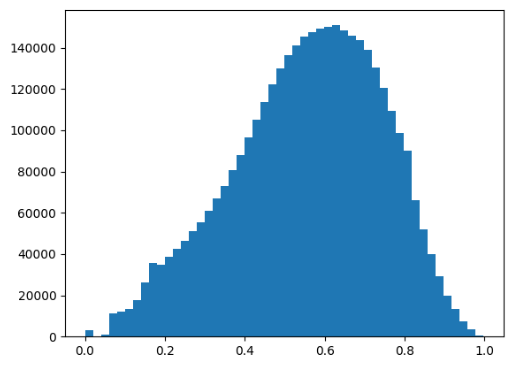
It is also important to know the distributions of each feature, information that Spotify stopped providing. However, other datasets can be used that contain listening and audio feature data of Spotify tracks. One of these, The Music Streaming Sessions Dataset from the Spotify Sequential Skip Challenge [7], contains audio feature data of 3.7 million tracks that were listened to over a period of a few months. The histograms of all eight features can be found in figure 2.1. Note that the dataset also contains additional features, but these are not available through an API.

Some of these features are distributed relatively evenly: *danceability*, *energy*, *valence*. These features will likely be good at representing music taste, as there is a wide variety. The other features, especially *instrumentalness*, have a single peak. This means that if a track has a value in this peak, there is no useful information, as it is very likely to fall in the peak already. Of course, these values can be useful if a track falls outside a peak.

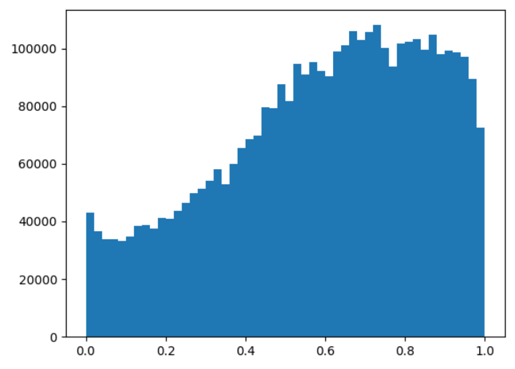
Since these audio features are all scalar values, they can easily be used with a similarity metric to calculate how closely related two or more tracks are. The assumption then is, that a track which is similar to a track that is enjoyed by a person, will also be liked by that same person.



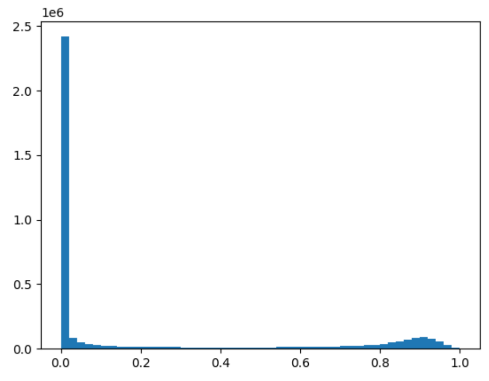
(a) Acousticness



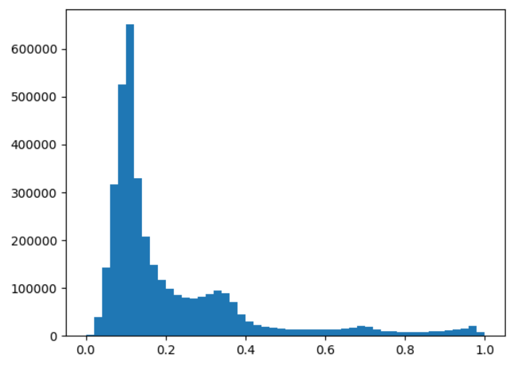
(b) Danceability



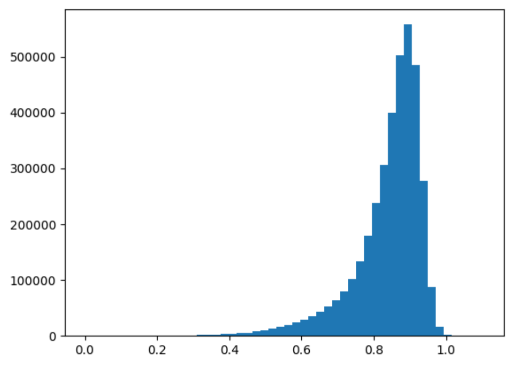
(c) Energy



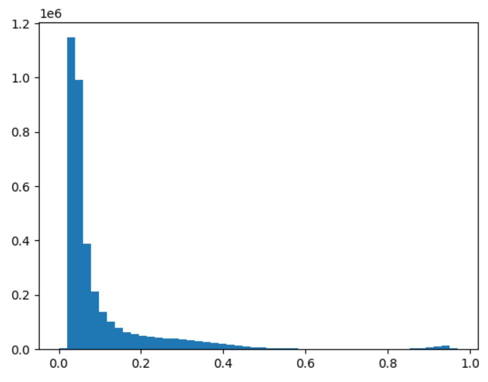
(d) Instrumentalness



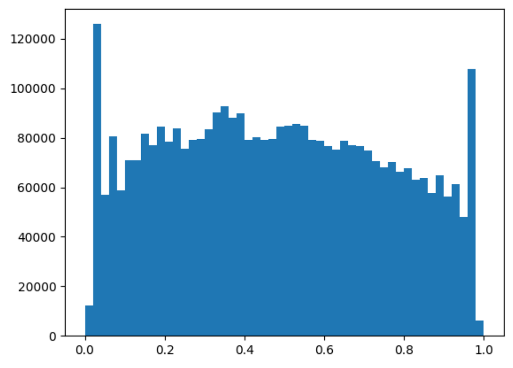
(e) Liveness



(f) Loudness



(g) Speechiness



(h) Valence

Figure 2.1: Histograms of audio features.

3 Experiment 1: Estimating Music Taste Based on Historical Data

In most cases, group recommenders require users to state their preferences explicitly. This task of stating preference usually consists of asking people to fill in some items they like or to rate some, or all, items given by the system. Asking this directly allows the system to quickly get a general overview of their tastes. However, this step can also be time consuming and even difficult for users in some cases. Section 2.2.2 listed some examples of systems that uses implicit methods of gathering the taste of their users, but research is lacking on implicit data gathering for group recommender systems. This section outlines an experiment that will give an answer to the following research question: **RQ1)** How can historical data be used to improve group recommender systems in the music domain?

3.1. Reasoning

As the introduction of this chapter stated, stating preferences explicitly can be both time consuming and difficult. This is especially true with systems for which it is unreasonable to ask a user to rate all the items, such as the music or movies domain. In these domains, a further complication is that the taste of a person is generally so varied that it is tough to represent their taste with just a few items. Finally, the time it takes for a system to generate a recommendation is sometimes important. The music domain can once again be used as an example, since it is unlikely that a group will want to fill in a long survey if they want to listen to music for some time as a group. In other words, the time it takes for the system to generate a recommendation might ultimately decide on if it is considered useful.

The benefits of using implicit data collection are obvious, since they directly counter the downsides of explicitly asking. The implicit method will generally be faster and easier to use, since there is no need to ask the users for their taste. However, it does of course come at the cost of always losing some performance, since a system will have to make some assumptions or estimations when collecting data this way.

The biggest question for implicit data collection is through which method is used to estimate the taste. INTRIGUE [1] used demographics to automatically classify people in different subgroups, while Flytrap [12] hooked in to a music player to keep track of what each user is listening to. Since music taste is so varied, the latter approach is more reasonable when creating a group recommender in the music domain. In order to remove the time investment required by Flytrap, it might also be possible to use historical data to estimate the taste of a person.

3.2. Design and Setup of the Experiment

Since this experiment compares historical data and explicitly given data, a platform needs to be used that facilitates both. The best candidate for this is a web survey, in which people are asked to fill in some tracks that fit their taste. Meanwhile, the web survey collects historical data using the Spotify API in the background. Since this experiment is relatively simple, no active supervision is needed. The web survey consists of the following steps:

1. Reading requirements and providing consent
2. Searching for and selecting tracks
3. Collect historical data

Step 1: Reading requirements and providing consent:
Upon visiting the survey, people are first given the requirements of participating (i.e. having an active Spotify account) in the experiment and an explanation of the experiment. This text also contains an informed consent form, further described in section 3.2.1. It should be noted that the explanation does say that *some* personal data is stored anonymously, but does not explicitly state that historical data will be retrieved. This was kept hidden to prevent scenarios in which participants alter their choices by either trying to avoid these tracks or by choosing them on purpose (e.g. “I will pick a track I listen to less because they already know my favourite tracks”). Upon agreeing with the terms, participants are asked to login with their Spotify account.

Step 2: Searching for and selecting tracks:
On the survey page, the task given to each person is: Please search for 5 to 10 songs that you would want to listen to. The page has a search function which allows to search on track, artist, or album. If a participant searches on artist or album, the page shows the top ten tracks of the artist or all the tracks from the album. Through either method, any resulting track can be moved in to the listen of selected items. Allowing artist and album search was done to make it easier for a person to explore the music space in an unfamiliar environment. When at least five tracks are selected, the submission button becomes active. Initially, this active state is shown by just an outline, which becomes fully filled in when ten tracks are selected. This choice was made to create a higher incentive for people to select ten tracks. Finally, there is an option for feedback, in which participants can explain their choices.

Step 3: Collecting historical data:
In the background, the code requests historical data from the logged in Spotify account. Specifically, the top listened to tracks and artists from the three time ranges (short, medium, long) can be requested. Since participants can fill in a maximum of ten tracks, the API requests were also limited to ten items. In some cases, for example when a participant use Spotify infrequently, it is possible that the API call returns fewer or zero items for a combination. The retrieved items are stored together with the selected tracks in a fully anonymous user profile upon submission

3.2.1. Informed Consent
Participants were recruited on a voluntary basis via social media and by asking friends and family. While participation was anonymous, it is likely that most participants are based in Europe due to social circles. No compensation for the participation was given, monetary or otherwise.

Since no personally identifiable data is stored, an opening statement is sufficient to ask for informed consent. This opening statement can be found in appendix A and is based on a template offered by the TU Delft [15]. This experiment was cleared by the TU Delft Human Research Ethics Committee.

3.2.2. Technical Details
The web survey was built using PHP, using the Symfony framework on the back-end. This choice was made fairly arbitrarily as it was the most familiar platform. On the front-end, JavaScript was used in combination with jQuery and Bootstrap. The design of the survey was used, with permission, from a project found on GitHub[1]. Data was stored in JSON files after being processed and anonymised. The survey was self-hosted on a server in order to comply with the General Data Protection Regulation and the TU Delft Human Research Ethics Committee. The data collected from the experiment was processed with Python.

3.3. Data Sets
Since the collected dataset (further referred to as the “real” dataset) needs to be compared with other data, two other datasets were created. One consisting of tracks that were picked randomly and one

[1]`https://github.com/hereismari/spotify-flask`

that uses the Spotify recommender to collect tracks that should be similar to the ones picked by a participant.

3.3.1. Comparing Recommendations

The “recommended” dataset was created to compare differences between historical data and tracks generated by a recommendation algorithm. This recommender[2] can be seeded by up to five items, which all have to be either tracks, artists, or albums. In this case, the top five tracks per time range from the historical data are used as seeds. In total, three track lists are generated per person, each having the same amount of tracks as the historical data does. This dataset should score high in the metrics, since the tracks should be similar to the items that were used to generate the recommendations with.

3.3.2. Random Dataset

The “random” dataset will serve as the baseline, as the historical data needs to score higher than random data in order for it to increase the performance of a group recommender. Generating random data was done by using the search API from Spotify. In order to have a high entropy, several parameters were randomised. The initial query consists of a random letter (a-Z) and a time range consisting of a random year between 1950 and 2020 as lower bound, and a random year between the lower bound and 2020 as upper bound.

This query also allows for an offset, which can be used to go, as it were, to the next page. Due to limitations, only the first 1000 items can be retrieved. From these 1000 tracks, a random one is chosen. This process is repeated until each track list for a time range consists of the same amount of tracks as the person has. This method was adapted from a guide which has since been removed from the Spotify website.

This approach lead to 4.9% (150 out of 3210) of the songs occurring more than once. The expectation is that this dataset scores lower than the “real” or “recommended” datasets.

3.4. Evaluation

There are several ways the results of the experiment can be evaluated. The simplest method is to count the number of tracks or artists that match between the selected tracks and any of the datasets. Of course, due to the nature of the recommended and random datasets, it is unlikely that there will be any matches between those and the selected tracks. This comparison method serves mostly as a way to compare the three time ranges of the historical data.

Another method is to use a similarity metric to calculate a score for each track. Acoustic similarity has been used recently, for example by Cheng et al. [11]. This method can be used with all three datasets and can use the audio features mentioned in section 2.7.2. In order to calculate such a score for a given track, the selected tracks are used as a user profile. That is, a similarity score is calculated for each combination of the given track and each of the selected tracks. These similarity scores are then averaged to come to a single score, as given by equation 3.1. The values have a minimum of 0.0 (the tracks are very dissimilar) to 1.0 (the tracks are equal), but values are typically between 0.7 and 1.0. By averaging the scores of a track list, this method can be used to compare two track lists against each other. Any similarity metric can be used, but for this experiment cosine similarity will be used. Cosine similarity is widely used and is useful to calculate a score based on the contents of items [56].

$$\text{score}(u_k, t_i) = \frac{\sum_{t_j}^{u_k} \text{similarity}(t_i, t_j)}{n} \tag{3.1}$$

where: u_k = A given user
t_i = A given track
n = Number of tracks in a user profile

Finally, a t-test can be used to check if two track lists or datasets have a significantly different distribution. As the samples are independent, Welch’s unequal variances t-test [59] is used.

[2] https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-recommendations

3.5. Data Storage

Some anonymised data is stored for each participant. Like mentioned before, each participant is asked to select five to ten tracks from Spotify, the uri[3] of each track is stored. For each time range, the uris of at most ten "top" tracks and artist are stored. There is a field for feedback which is either an empty string, in case the participant did not leave feedback, or an escaped string with a maximum of 2048 characters with feedback. Finally, the Spotify id of the participant is hashed using the MD5 algorithm. This hash algorithm is suboptimal [25], but since there are no big risks associated with the user id of a participant being discovered it was decided that using this function is sufficient.

The data can be found at [8]. The other two data sets used can be found in the same place.

3.6. Results

In total, 88 people participated in the experiment. As expected, some data is missing since a small amount of participants use Spotify sparingly. The completeness can be seen in table 3.1. On average 8.6 tracks were selected. Most participants had ten items in each time range, with 77, 80, and 87 participants having ten tracks for respectively the short, medium, and long time range. 24 participants left feedback, with comments usually consisting of an explanation why they picked the tracks or simple saying they enjoy their selection. There was also feedback stating that it was difficult to come up with enough tracks, possibly due to question asked being fairly broad and the participants were only given a vague description as inspiration. Initially, the minimum of selected tracks was ten, but was lowered to five when this was observed during testing.

Table 3.1: Data completeness of the first experiment, the maximum number per category is 880.

	Short	**Medium**	**Long**
Artist	650 (73.9%)	771 (87.6%)	855 (97.1%)
Track	771 (87.6%)	807 (91.7%)	877 (99.7%)

3.6.1. Matching Tracks and Artists

The average of matching items between the historical and selected data can be seen in table 3.2, box plots can be found in figure 3.1. The short and medium time ranges have a better result compared to long term. A potential cause for this is that long term contains tracks or artists that someone has listened to the most for (potentially many) years. This does not necessarily represent what someone wants to listen to right now. Taking all time ranges into account, it is unlikely that someone picks the exact track that they have been listening to, either recently or longer ago. It is more likely that someone picks an artist that is also in their top artists. The reason for this is unknown. It could either be because it is more likely to pick a favourite artist over a song, or it could also have more abstract reasons, such as people wanting to have some variety. As the available logs or feedback fail to provide a reason, more research is needed figure out this reason.

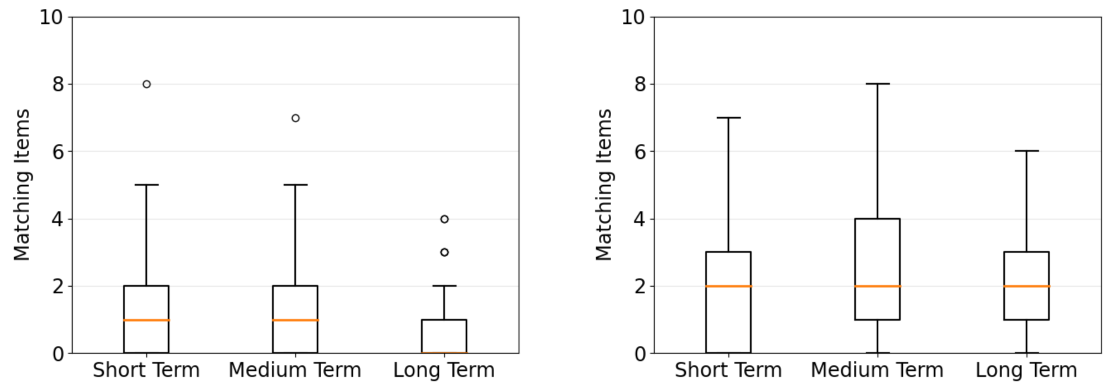
Table 3.2: Means and highest outliers of the number of matching items

	Tracks			**Artists**		
	Short	**Medium**	**Long**	**Short**	**Medium**	**Long**
Mean	1.23	1.25	0.66	2.05	2.40	2.11
Highest Outlier	8	7	4	7	8	6

3.6.2. Comparing the Datasets

The average scores for each dataset and time range can be found in table 3.3, box plots for each dataset can be found in figure 3.2. Similar to the previous result, the short time range has better results than the other time ranges. Furthermore, the real and recommended datasets have scores that are very similar to each other, while the random dataset scores lower. Since typical scores are between 0.7 and 1.0, the difference between scores maybe be larger than initially appears.

[3]e.g. `spotify:track:4uLU6hMCjMI75M1A2tKUQC`



(a) Matching tracks

(b) Matching artists

Figure 3.1: The number of items that appeared in both the historical data and selected tracks per time range.

In order to check if the datasets are significantly different from each other, SciPy's implementation of Welch's t-test[4] was used on each pairing of datasets (i.e. real and random data, real and recommended data, and recommend and random data). In this case, the null hypothesis is that two independent samples have identical means. This means that two datasets are significantly different if the p-value of the t-test is under 0.05. The results of the t-tests can be found in table 3.4. Any test done with the random dataset gives p-values significantly lower than 0.05, thus it can be concluded that the distributions are significantly different. The p-values obtained by comparing the real and recommended dataset show indication that these datasets are related. This holds for the short time range ($t(770) = 0.937, p = 0.349$), medium ($t(806) = 0.486, p = 0.627$), and for long ($t(876) = 1.55, p = 0.120$). The lower p-value for the long time range in consistent with other results, since the long time range also has less mutual tracks and artists.

These results suggest that the real and recommended datasets are related and will be similarly enjoyed by listeners. In contrast, the random dataset is very likely to be unrelated to the other two datasets, suggesting that it will be enjoyed less.

Table 3.3: Average scores per dataset and time range.

	Short	**Medium**	**Long**
Real	0.8870	0.8839	0.8859
Recommended	0.8837	0.8820	0.8803
Random	0.8480	0.8459	0.8475

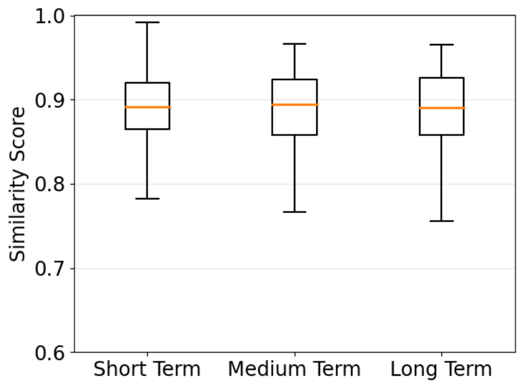
Table 3.4: Result of the runs using Welch's t-test.

	Short (df = 770)		**Medium** (df = 806)		**Long** (df = 876)	
Data sets compared	**t**	**p**	**t**	**p**	**t**	**p**
Real vs. Recommended	0.937	0.348	0.486	0.627	1.55	0.120
Real vs. Random	8.72	7.75e-18	8.04	1.86e-15	9.11	2.46e-19
Recommended vs. Random	8.14	9.25e-16	7.76	1.65e-14	7.52	8.36e-14

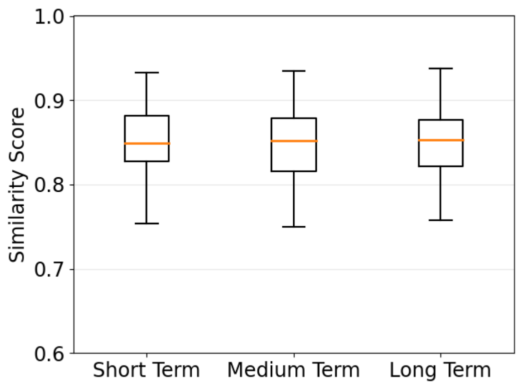
3.6.3. Feedback Analysis

Since providing feedback was optional, only 24 people (27%) made use of this option. Most feedback explains in one sentence why the participant picked their songs, ranging from calling their songs "good shit" to just saying the choices represent their music taste. Some people left more elaborate feedback, with one participant even explaining all songs individually. One common reply is that their choices represent their overall music taste, which was not what the experiment asked for. This shows that

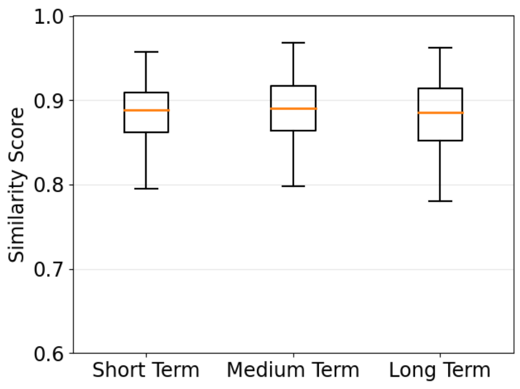
[4]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html`



(a) “Real” dataset



(b) “Random” dataset



(c) “Recommended” dataset

Figure 3.2: Box plots of the distances as calculated using the similarity score method for each time range and data set.

the task was potentially confusing for some. Finally, some participants expressed that it was difficult to think of five tracks on the spot and with no inspiration. This feedback was provided in person or through messages, mostly during testing. This is consistent with one of the earlier stated issues of explicit data collection.

4 Experiment 2: Design and Setup

While the setup of the first experiment was fairly straightforward, the second experiment is more involved. In order to answer **RQ2)** and **RQ3)**, a group recommendation algorithm needs to be implemented and used and a questionnaire needs to be offered to groups participating in the survey to get data about the recommended playlists. Finally, the survey should also provide a way for groups to be formed. It was impossible for groups to meet up to participate in the experiment, so all stages of the survey should be able to be performed asynchronously. This chapter will describe the design of the group recommender and the setup of the experiment, beginning with some background information on why this experiment was performed. Then, it will explain the group recommendation design and the choices that were made during this stage. The survey will be described in the final section of this chapter.

The data obtained from this experiment will answer the following questions:

- **RQ2)**: To what extent does the performance of a group recommender system vary when considering different aggregation strategies?
- **RQ3)**: To what extent can the acoustic similarity of playlists, recommended for a group, be measured and increased?

4.1. Background

Most research using a group recommendation algorithm often chooses to use a single aggregation strategy for their system design or experiments. Of course in most cases, a justification is given for the choice. However, there is usually no comparison in the same experiment between different aggregation strategies. Of special note is *Probability Weighted Sum* and the variants, which appear in experiments in the music domain with little motivation behind the choice. This lack of exploring different options also occurs in other dimensions of group recommenders. There is no research comparing the performance of active versus passive groups for example.

There are also different aspects that need to be explored more, especially in the music domain. Specifically, most systems either ask group members to rate all items (e.g. radio stations in MusicFX [33]) or use metadata to estimate ratings (Flytrap [12]). By using audio features as way of estimating ratings, the viability of these are also tested.

Finally, social effects are often ignored in experiments. In some cases, when real data is used, the people belonging to a group are little to not familiar with each other. When synthetic data is used, social effects cannot occur at all. The conditions of the experiment and groups being unable to meet up and discuss the track lists do limit testing in this regard. Nevertheless, there should still be some social effects that can be discovered in the data.

4.2. Group Recommendation Design

Since the second experiment plans to test a group recommender in a real-life scenario, a system needs to be build. As this system will be in the music domain, not all steps mentioned in section 2.2.2 are relevant. Specifically, the last step, helping the group reach consensus, is unnecessary, and for the third step, presenting the recommendation, the system will not have to explain the recommendation for simplicity reasons. The first two steps are of course important for the group recommender and will be outlined in this chapter. The dimensions of this system, previously described in section 2.2.1, will also briefly be discussed.

4.2.1. Gathering Data

As described before in section 3.1, data collection can either be done explicitly by asking people to state their taste or rate a number of items, or implicitly. The same section showed the benefits and downsides of either method. The first experiment also showed that historical data outperforms randomly selected items. Nevertheless, the results were inconclusive and untested in a group setting.

The latter is especially problematic, since it is likely that a person will express a different taste for a group that came together with a specific purpose. For example, a person choosing songs to listen to while studying as a group is unlikely to chose very distracting music, even if that would be their preference. This means that goal of this stage should not be to capture the overall taste of a person, but rather their taste within the context of their group. Requiring groups to have a purpose in mind while selecting the songs additionally aids the group members in picking tracks they think are enjoyable for the group as a whole [43]. To accommodate this, a mix of implicit and explicit data collection is used. The group recommender still asks each person to select five tracks based on the purpose of the group, but pre-fills this list with the top five track in the short term time range, as returned by the Spotify API. This allows users to still search for their own tracks, but also offers an initial selection that was automatically selected. This also counters feedback from the first experiment, where participants stated it was difficult to select tracks on the spot. Since all selected tracks are retrieved or found using the Spotify API, so they are guaranteed to have the necessary audio feature data available.

4.2.2. Calculating Similarity Scores

Since the music library is as big as it is, it is impossible to ask participants to rate all items. This means that the group recommender will have to estimate scores for the items that are new for the user. This can be done through either collaborative filtering or content-based filtering. Both are viable in the music domain, and, while collaborative filtering is currently more widely used, content-based filtering can also show strong results. The deciding factor was the difficulty of implementation. Collaborative filtering needs a lot of data, which is not directly available, and time in order to make and train an accurate prediction. Content-based filtering does not have the same issues and will be used for the system.

Content-based filtering is used with an approach similar to the one described in section 3.2, which uses the audio features to calculate similarities between tracks. [11] showed that there is a connection between acoustic similarity and recommendation and that acoustic similarity is consistent across different people. Since their research used the same audio features as this experiment, it is a reasonable assumption they can be used to estimate taste as well.

The algorithm uses the five selected tracks selected by a user to generate a taste profile. These tracks are given the highest score (1.0) and are used to calculate the similarity scores of the other tracks. In order to then estimate a similarity score for a given track, a similarity metric is first used to calculate the individual scores for that track with each of the selected tracks. These similarities are then averaged to get to a single similarity score, as can be seen in equation 4.1. This is the same as equation 3.1, but used for a different purpose. By calculating this score for each of the tracks selected by other members of the group, a ranking of tracks is found for that person. This approach assumes that the closer a track from another user is to their own selected tracks, the more they will enjoy that track.

$$\text{similarity\_score}(u_k, t_i) = \frac{\sum\limits_{t_j \in u_k} \text{similarity}(t_i, t_j)}{n} \tag{4.1}$$

where: t_i, t_j = A track
u_k = A user, containing a set of tracks
n = Number of tracks in a user profile

4.2.3. Implementation of Aggregation Strategies

After the group recommender has obtained and estimated a similarity score for each item and person, it has to aggregate the scores to be able to recommend a track list. Out of the possible aggregation strategies, the three mentioned earlier in section 2.4, *Probability Weighted Sum*, *Fairness*, and *Least Misery*, will be used.

For the implementation for *Probability Weighted Sum*, the variant first described by MusicFX [33] is used. That is, for each track, all similarity scores are first squared and then summed together. Squaring the scores improves the performance as shown by [30]. The score for each track is then normalised so that all tracks together sum up to one. This can be seen in equation-form in equation 4.2. The algorithm then randomly picks ten items, using the resulting values are then used as weights.

$$p(t_i) = \frac{\sum\limits_{u_k \in U} \text{similarity\_score}(u_k, t_i)^2}{\sum\limits_{t_j \in T} \sum\limits_{u_k \in U} \text{similarity\_score}(u_k, t_j)^2} \tag{4.2}$$

where: t_i, t_j = A track
u_k = A user
T = The set of all tracks
U = The set of users

The *Fairness* strategy randomly selects a member of the group and picks their highest scored item. In the context of this group recommender, this means that the items chosen are always one of the items the person chose themselves. This selected item is then also removed from their available items. The next user is then given a turn by picking their top item, looping back to the first user if needed. This continues until ten items are selected.

Finally, *Least Misery* works by first applying a `min` function over the similarity scores for each track, which is then sorted in descending order, as seen in equation 4.3. This results in a list with the highest "worst" similarity scores being selected first. Once again, the top ten items of this list are selected for the playlist.

$$\texttt{least\_misery}(t_1, ..., t_n) = sort(min(t_1, ..., t_n)) \tag{4.3}$$

Regardless of the aggregation strategy, the algorithm returns a list of ten recommended items. The group recommender generates one playlist per aggregation strategy and presents the three resulting playlists to the members of the group. While section 2.3 showed that sequences of items should have certain qualities, such as a strong narrative flow and a strong ending, the decision was made to have the system not account for these qualities. This keeps the aggregation strategy as the only free dimension and ensures that the ratings obtained from the survey actually give an indication of the quality of the strategies.

4.2.4. Adding More Serendipity?

This new playlist generated by the algorithm can be seen as the preferences of a single user. This means that the tracks can be fed to a new recommendation algorithm to get results that are more serendipitous than the original tracks. This approach has benefits and downsides.

The obvious benefit is that people will likely get songs similar to the ones they like, but might be unfamiliar. This makes the session more surprising and potentially more entertaining. It can also help with making choices less polarising, since in tests with using the Spotify recommender API, it appeared that the recommender favoured more popular tracks.

A downside is that the tracks people picked will not appear in the playlist, which could disappoint people. If a person is asked to pick tracks, their expectation will be that these songs appear in the resulting track list. As a compromise, it is possible to combine the selected and recommended tracks. However, as multiple track lists need to be compared with each other, they are relatively short at ten tracks each. The issue of tracks not appearing in the recommended track lists will not be solved by this method.

Another downside is that publicly available recommenders are black boxes, meaning the way they operate is not publicly known. In the case of the Spotify recommender, this is problematic since it is not known whether their system takes more in to account other than the items given as a seed. This has the potential to lead to unpredictable results, which would make it more difficult to reach an objective statement of a recommendations. These issues are enough to show that adding serendipity in this specific case is undesirable; the group recommender will only use the selected tracks for the recommendation step. This does, of course, not mean that adding serendipity is unfavourable in every case.

4.2.5. Group Recommender Dimensions

The selected options for each dimension were largely influenced due to the way the experiment had to be set up. In fact, most options were unavailable, since it was impossible for groups to meet up and discuss the playlists due to the COVID-19 pandemic. This change the set up of the experiment drastically, changing it to one that was as asynchronous as possible, in which it is unnecessary for the group members to have any interactions with each other, aside from forming the group.

This means that the group recommender works with **passive groups**, who cannot influence the score aggregation, and that the recommender items are presented as **the only option**. Furthermore, since the survey will be asynchronous, it is also important to make the process be user friendly. This is to encourage participation. One aspect of user friendliness is that the process as a whole only takes up a small amount of time, so the most suitable option is to have all preferences **known** by asking users at the start of the survey, rather than having the system learn over time. Since the experiment will compare track lists generated by different aggregation methods, the system will recommend a **sequence of items**. Finally, section 4.2.2 explained the choice for **content-based filtering**.

This leaves the system to have one degree of freedom, **the method of score aggregation**, which means the system can be used in the next chapter and experiment to compare different aggregation methods.

4.3. Experiment Setup

This section will outline how the experiment is set up, from the design of the survey and the accompanying website, the design of the group recommender, and how data was obtained and stored.

4.3.1. The Survey

The entire process consists of several steps:

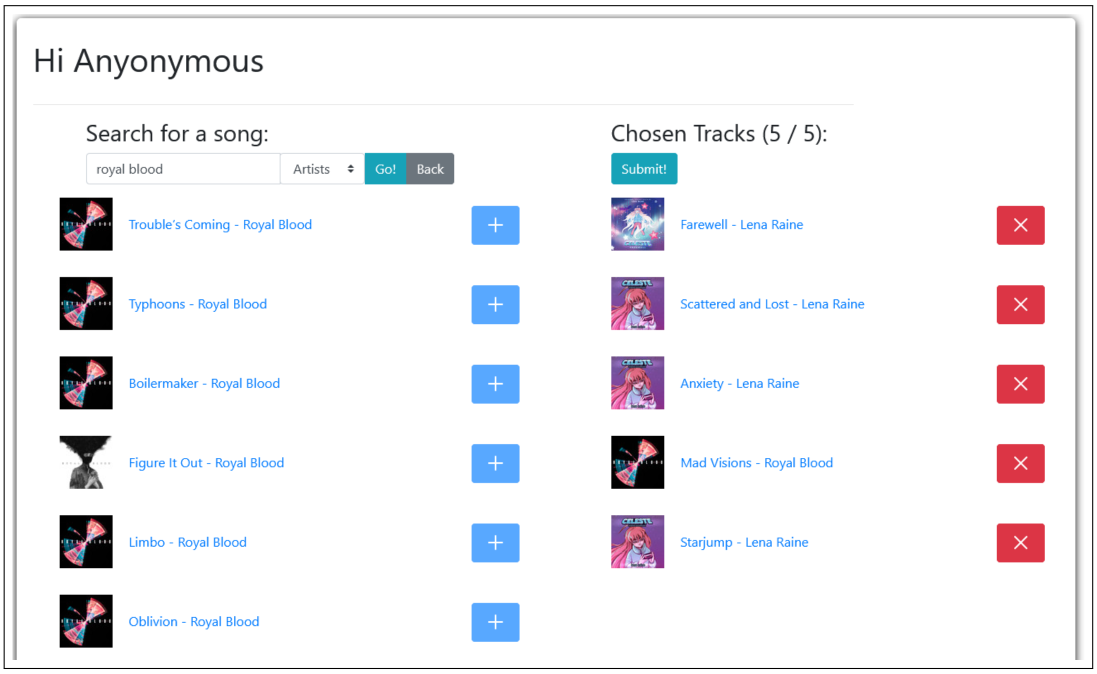
1. Group creation
2. Data collection
3. Filling in the survey

These steps roughly follow the steps listed by Jameson and Smyth [23] in that the individual preferences are acquired in the “Data collection” step, the recommendations are generated between that step and the questionnaire, and the recommendations are presented in this final step. Due to groups being unable to meet up in real life, these steps were designed so that each participant could work on them in their own time. The session can only progress to the next stage when every member of the group has completed the current step.

Step 1: Group Creation

For the group creation step, the “leaders” of the groups were found by asking friends and family directly and by asking them to spread the word to their social circles. These people were asked to first come up with a purpose for the group, for example *working out* or *studying*[1]. This purpose is used as inspiration for the participants to come up with tracks for the group, as activities or purposes are a common reason to organise a playlist [13]. After filling in such a group purpose, there were then asked to invite two to seven people to join their session, by filling in their email-address. This means that in most, if not all, cases, the group members were familiar with each other and the preferences of the other group

[1]Of course, other purposes were allowed as well.


Hi Anyonymous

Search for a song:

royal blood | Artists | Go! | Back

- Trouble's Coming - Royal Blood
- Typhoons - Royal Blood
- Boilermaker - Royal Blood
- Figure It Out - Royal Blood
- Limbo - Royal Blood
- Oblivion - Royal Blood

Chosen Tracks (5 / 5):

Submit!

- Farewell - Lena Raine
- Scattered and Lost - Lena Raine
- Anxiety - Lena Raine
- Mad Visions - Royal Blood
- Starjump - Lena Raine

Figure 4.1: Screenshot of the data collection step of the web survey. The left column is used to search for tracks, artists, or albums. The right column shows the current selected items and has a button to submit the tracks.

members. Once everyone has joined the session, the group leader can advance the group to the next step.

Step 2: Data Collection
In the data collection step, all group members were asked to select five songs they would want to listen to with the group given the selected purpose. As explained in section 4.2, the five tracks (as retrieved from the short term time range) are presented as initial selection. While participants are searching for tracks, the website stores all the tracks that were returned from the search queries. Additionally, all tracks that were interacted (i.e. hovered) with were kept track of in a separate list. A screenshot of the data collection step can be found in figure 4.1. The session proceeds to the survey step once all members of the group have filled in their choices, at which the group recommender algorithm is invoked. An email notification is send to each person once the algorithm has recommend all three playlists.

Step 3: The Survey
In this step, all participants are presented with the generated playlists and are asked to listen to the playlists. In order to prevent survey fatigue from altering the results, each participant was presented the three playlists in a random order. Each playlist can also be exported to Spotify, to make it easier to listen to the entire playlist in one sitting. Initially, each participant was shown just the playlists and instructions to start the survey once they had listened to each playlist. The survey could be started by pressing a button, which presented the participants with the first set of questions.

This survey asked questions in three categories per playlist, as given in Table 4.1: whether the tracks were enjoyable (“Like” rating), whether the participants were happy with the inclusion of their own tracks (“Selection” rating), and whether the recommendations fit the purpose (“Suitable” rating). A five-point Likert scale was used to let participants answer these questions. Aside from general ratings for the entire playlists, there was also an option to rate each track individually for each category, with the middle (third) option set as default. For the general rating, no default option was set. Finally, there was also an option to leave feedback at every question and at the end of the survey, leading to 28 questions in total. It should be noted that each question also had an accompanying description, which explained the question further. This also helped create a distinction when the actual questions were the same.

Table 4.1: Specific questions asked in the survey for each category. Each of these questions was asked for every playlist, resulting in 28 questions in total.

Type		Question Asked
Like		
	Overall	How much did you enjoy this playlist?
	Specific	How happy are you with each of your chosen tracks?
	Feedback	Any other feedback for how much you enjoyed this playlist?
Selection		
	Overall	Are you satisfied with the position of your chosen tracks in this playlist?
	Specific	How happy are you with each of your chosen tracks?
	Feedback	Any other feedback about your chosen tracks?
Suitable		
	Overall	Do you think this playlist fits the purpose of the group session <purpose>?
	Specific	Do any of the songs fit this purpose especially well or poorly?
	Feedback	Any other feedback about how well this playlist fits the purpose?
General		Do you have any other feedback?

4.3.2. Informed Consent

As with the first experiment, participants were recruited on a voluntary basis via social media and by asking friends and family. All participants were based in Europe, with a focus on the Netherlands. No compensation for the participation was given, monetary or otherwise. To invite people to join a group, their e-mail addresses were used and securely stored before asking permission to use the private data. In the invitation to join the group, an option was given to immediately delete all personal data.

Contrary to the first experiment, where no personally identifiable data was stored, this experiment does store such data. For this reason explicit consent is required from each participant. Participants were asked to give consent before officially joining the group, they could also request to have their user data deleted at this point. The statements and questions, based on a template from the TU Delft [15], can be found in appendix A. This template originally requires a signature, but it was impractical to ask a physical signature for every participant. Instead, a final checkbox asking for consent was put in place. In combination with an email address this is also an identifiable signature that is valid for consent and record keeping [58].

4.3.3. Technical Details

The back-end of the server was made using PHP and the Laravel framework, the front-end uses Vue and Bootstrap. Laravel and Vue were used over Symfony and jQuery as they are easier to develop with and, especially in the case of Vue, less likely to result in bugs in the survey. For the survey element specifically, the survey.js library was used. The surveys can be made on their website and imported into code directly as a JSON object. MongoDB was used to store data. MongoDB is a NoSQL-type database, which is better at storing variable length records than SQL-type databases. The survey and database were self-hosted for the same reasons in the first experiment: to comply with the General Data Protection Regulation and the TU Delft Human Research Ethics Committee. Python was once again used to process the data after the experiment had concluded.

4.3.4. Data Storage

Each group session has a randomly generated session id and the purpose of the group. The generated recommendations are also stored here, with each track list containing a list of Spotify track data[2]. Each track list also contains metadata, such as the aggregation strategy used for the recommendation and the estimated ratings for each group member.

The session have a one-to-many relation with the users. For each user, their Spotify user id, their email address, and session id they belong to are stored. Both of these are used to allow for the user to return to the session at a later time, allowing the session to be completed asynchronously. While a user is selecting tracks, the items they select, interact with, or saw, are kept track of and stored separately.

[2]As retrieved using `https://developer.spotify.com/documentation/web-api/reference/tracks/get-track/`

Finally, the answers to the survey for each user are also stored here, which is stored directly as the object returned by the survey library.

The data of this experiment can be found at [8], all personal data is anonymized.

5 Experiment 2: Comparing Different Aggregation Strategies

The original purpose of the second experiment is to compare different aggregation strategies. While this topic is popular in the field of group recommendation, most often it is done using synthetic data, which has been shown to be inaccurate due to artefacts that commonly appear [30]. There are also multiple psychological effects that only occur with groups in real-life settings, section 2.5 went into more detail on this. This means that comparing aggregation strategies will only be accurate when done in a real-life setting, which is infrequently done in research. The web survey outlined in the previous chapter was built around this purpose, allowing for multiple playlists with different aggregation strategies to be recommended at the same time. The questions asked in the survey also help to compare the different strategies. This chapter will aim to answer **RQ2)**: To what extent does the performance of a group recommender system vary when considering different aggregation strategies?

5.1. Results

During the run of the experiment, 40 people participated in the survey, spread over eleven group. An additional three people were unable to complete the survey, but did select their tracks, which means their tracks were used for the recommendations.

Group statistics

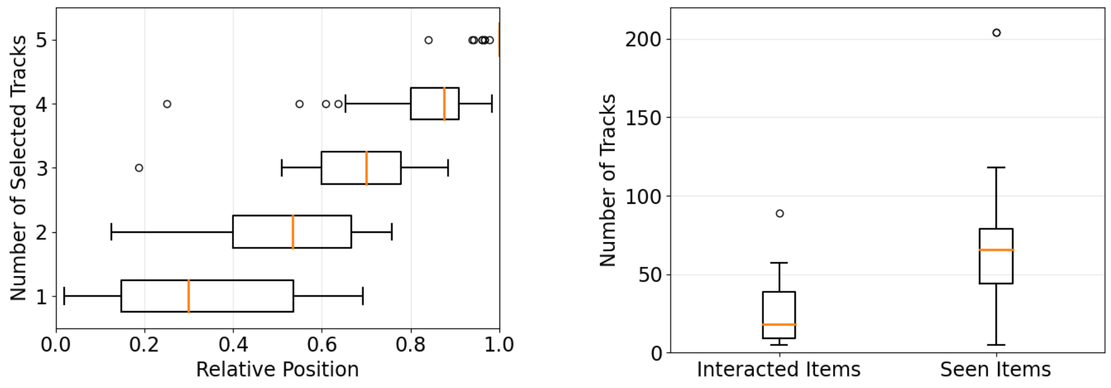
The eleven groups had a median of four people per group, with one group having an outlier of six people. This group also took, by far, the longest to complete, with each step of the experiment taking several weeks to complete. Accurate timestamps are unavailable, so the time it took each group to complete the entire process can only be given with anecdotal evidence. As written in section 4.3.1, it can be assumed that every person in a group knows all other group members and has at least a broad idea of their preferences. However, it is unclear if participants actually took the preferences of other group members into account.

The purposes of the group, as given by the group leaders, can generally be summarised as “having fun with friends”. The group purposes neatly line up with events and activities that are commonly used as starting point for new playlists [13]. It should be noted that due to most groups being unable to meet up while the experiment was running, participants were told that imagining the situation or purpose was acceptable. The full list of group purposes can be found in appendix C.

Track statistics

A total of 210 unique tracks were chosen by the 43 participants. They interacted with (i.e., hovered over and/or selected) 805 unique tracks, and 2162 unique tracks were retrieved as part of the search queries. Each person had an average of 2.6 songs per playlist; only in rare cases did a playlist not contain any tracks a person selected. Regardless of the used aggregation strategy, this average holds, but the variance is larger for *Probability Weighted Sum* and *Least Misery* than for *Fairness*. It should be noted that the design of the first two strategies does not ensure that all group members had at least one of their own tracks in the playlist. On average, participants saw 68 different items while selecting their tracks and interacted with 25. There were no signs of survey fatigue, a more elaborate analysis of this can be found in section 5.4.

Figure 5.1 shows the box plots of the positions of when participants on average found their tracks, relative to the full sequence of items they interacted with. This data shows that most participants settled on their tracks in the second half of searching for items. The box plot shows that the initial pre-filled selection of tracks that the person recently has listened to most, was used by roughly half of the participants. A total of twenty used at least one item from the historical data, while four used all five and did not search for other tracks. This shows that using historical data as initial selection was likely useful to serve as a starting point. The same figure also shows the average session length in both the total number of items seen while selecting items and the number of items that were interacted with.



(a) Relative positions of when tracks were selected

(b) Average Session Length

Figure 5.1: Box plots of the (relative) positions of when the tracks were selected in the data collection step and of the average number of interacted and seen items.

Survey Ratings

The average survey ratings per playlist and per survey category can be found in Table 5.1. *Probability Weighted Sum* scored slightly higher in the “Like” category compared to the other two strategies, but the ratings also have a larger standard deviation. For the other two categories, *Fairness* scored higher than *Probability Weighted Sum* and *Least Misery*. Due to common feedback that the “Selection” questions were confusing, especially when tracks were not in the playlist at all, it will be excluded from further analysis.

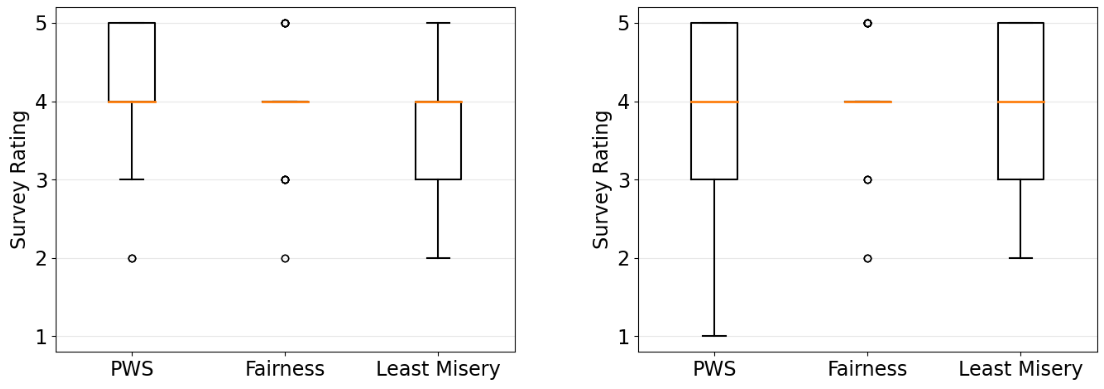
Figure 5.2 shows box plots for both the “Like” and “Suitable” categories and each aggregation strategies. This figure confirms that the results for the *Fairness* strategy vary less compared to the other two. Since this strategy also has a more consistent number of tracks per person that end up in a playlist, there likely is a correlation between tracks being selected by a person, and higher ratings.

Table 5.1: Means and standard deviations of the survey ratings given to the playlists for each question.

	Like		**Selection**		**Suitable**	
Aggregation Strategy	**M**	**SD**	**M**	**SD**	**M**	**SD**
Probability Weighted Sum	3.98	0.86	3.55	1.11	3.88	1.09
Fairness	3.92	0.76	3.90	0.90	3.98	0.80
Least Misery	3.85	0.77	3.67	1.14	3.92	0.83

5.2. Individual Ratings

This correlation can be further examined by using the ratings given to individual tracks. While giving a 5-point explicit Likert scale rating to all individual tracks was optional (a rating of 3 was pre-filled by default), most participants did fill in individual ratings for the tracks. 82% and 75% of the individual ratings in the “Like” and “Suitable” category respectively had a rating that was changed from the default.



(a) “Like” category (b) “Suitable” category

Figure 5.2: Box plots of the ratings obtained from the survey. Both the ”Like” and ”Suitable” categories are shown.

A rating that was not changed can either mean the person did not want to fill in a rating, or that the default was an acceptable rating for that track.

The individual ratings can be divided in two groups: those given to tracks that were selected by the person answering the survey (referred to as “selected items”), and to tracks that were selected by others in the group. Comparing the average ratings for these groups shows that the ratings of the selected items are 0.94 higher in the *Like* category and 0.56 higher in the *Suitable* category. This is an expected result and consistent with observations from [16]. More surprising is that tracks directly following the items chosen by a person are also rated higher than expected. A visualisation of what tracks are meant can be found in figure 5.3, the green coloured items are the tracks that show this behaviour. This effect only occurs in the *Like* category and not in the *Suitable* category. This effect is likely an example of *assimilation* [31]. Figure 5.4 shows the differences between the selected and not-selected items and between items following a selected item and the others[1]. Using a t-test shows that this difference is significant ($t(238) = 2.397, p = 0.017$) when looking at all ratings. While it would make the most sense for this effect to happen in playlists using *Fairness*, it is actually absent in the *Fairness* aggregation strategy.

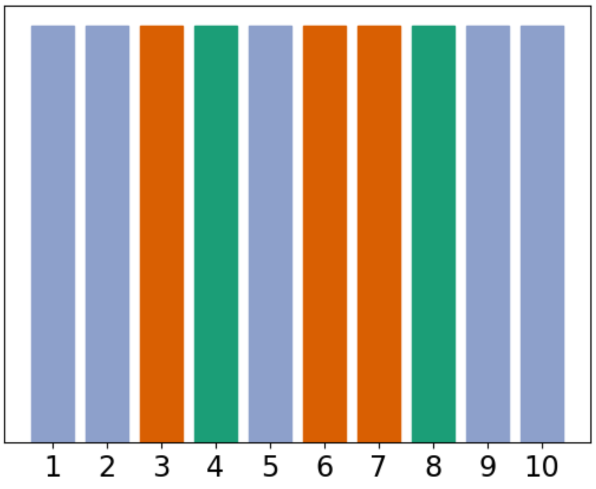
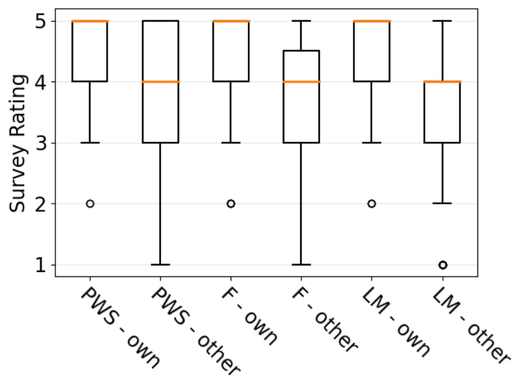
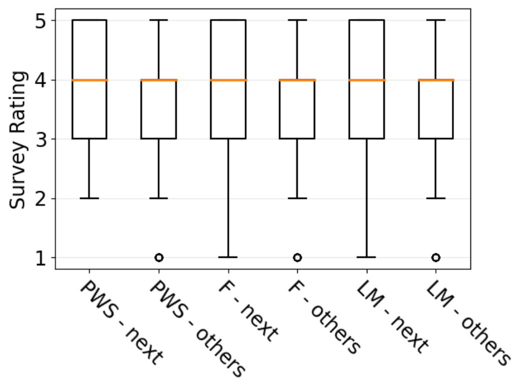


Figure 5.3: Example of which specific tracks and their ratings are taken into account when comparing the “tracks directly following a selected item”. Red denotes selected tracks, green the tracks that are taken into account, and blue all the other tracks. Note that the third red track directly follows another selected item, which means the accompanying rating is not used.

[1]To prevent bias, user-selected items that directly follow another user-selected item are left out of analysis.



(a) Selected items vs. non-selected items



(b) Non-selected items following a selected item vs. other non-selected items

Figure 5.4: Box plots of the ratings obtained from the survey divided in two ways. The first divides in selected and non-selected items. The second divides on items directly following a selected item and the other non-selected items.

5.3. Feedback

A total of 25 people left feedback in any of the available places. The majority of feedback given was positive and commented that the playlists were enjoyable. Most negative feedback focuses on a song being out of place given the theme of the group, for example: “Another really strong playlist, only one song I wasn’t keen on and the rest were good fits”. The group recommender algorithm actually ignored the reason of the group for the recommendations, rather, it was used for participants to have a starting point in selecting their songs. Similar to this, some feedback commented on narrative flow of the playlists. Making sure the playlists had good narrative flow was out of scope for this experiment, which sometimes resulted in harsh transitions between tracks.

Another common complaint was about the number of tracks that were selected for a playlist. Due to design of *Probability Weighted Sum* and *Least Misery*, giving all group members (roughly) the same number of tracks was not taken into account. This resulted in some cases where a person would have only one or even zero tracks in a playlist.

5.4. Survey Fatigue

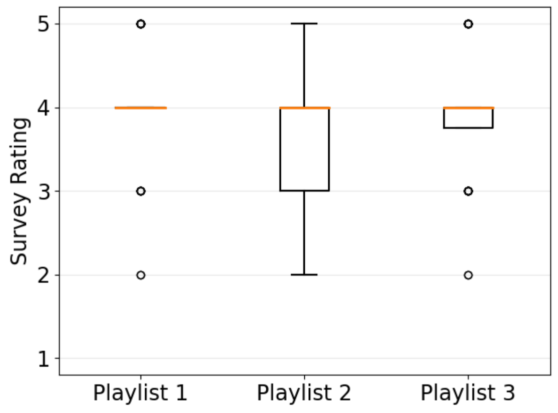
To combat survey fatigue, the playlists were given in a random order to each user. The correct order was stored as well, which makes it possible to reconstruct the intended order of playlists. The randomisation was done by swapping the positions of two randomly chosen playlists three times. Table 5.2 shows the amount of times each aggregation strategy was in any position.

Table 5.2: Number of times each aggregation strategy occurred in either of the randomised positions.

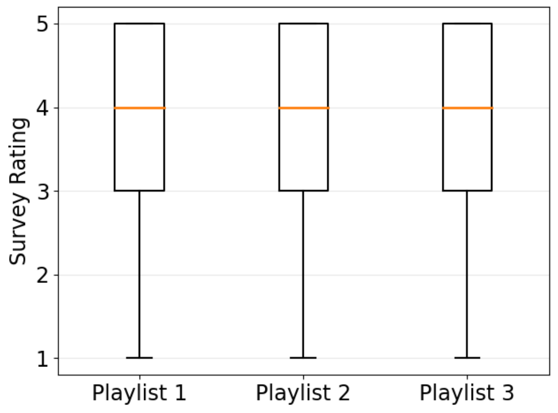
	Playlist 1	**Playlist 2**	**Playlist 3**
Probability Weighted Sum	17	9	14
Fairness	9	21	10
Least Misery	14	10	16

To see if survey fatigue still occurred, the ratings for each playlist (in the original order and the reconstructed order) were checked to see if the ratings dropped at the end of the survey. Box plots of these ratings, for both the overall playlists and the individual tracks, can be found in figure 5.5. These box plots show that the ratings are quite stable when comparing the playlists in the randomised order.

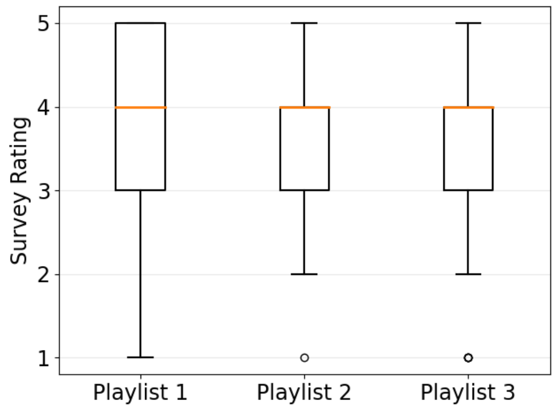
This is an indication that survey fatigue did not occur in most cases. However, the individual track ratings do show a potential sign of some survey fatigue. The limited survey interaction data shows whether a track rating was interacted with. Since the order of questions for each playlist was not randomised and always in the order of “Like”, “Selection”, and “Suitable”, there was a decrease in interaction over time. Section 5.2 showed that 82% of the individual tracks ratings in the “Like” category and 75% of the ratings in the “Suitable” category were interacted with. Since no further interaction data is available, it is impossible to know why these ratings were not interacted with, but it could be a sign that some survey fatigue occurred.



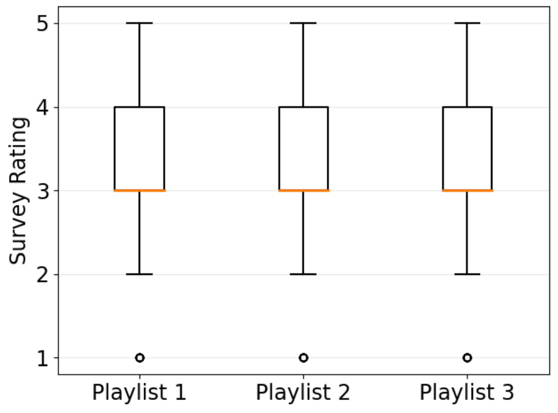
(a) Overall “Like” rating



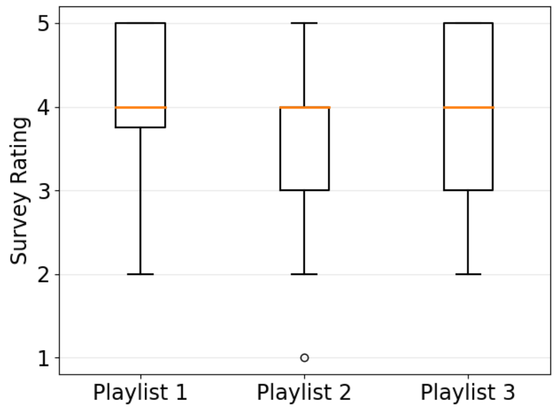
(b) Individual “Like” ratings



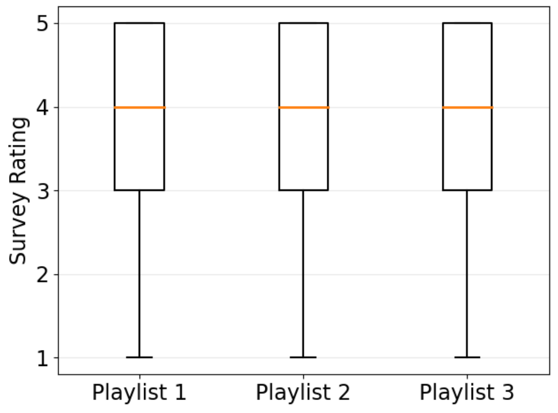
(c) Overall “Selection” rating



(d) Individual “Selection” ratings



(e) Overall “Suitable” rating



(f) Individual “Suitable” ratings

Figure 5.5: Box plots of the ratings given to each playlist in the randomised order.

6 Experiment 2: Acoustic Similarity

Most group recommender systems in the music domain collect user preferences in one of two ways. Either the collection of items is small enough to make it feasible for a person to rate all items, or the system uses some form of metadata to create links between items and infers preferences through those links. The group recommender from chapter 4 used the second method, since a similarity score was calculated using the tracks participants selected as a user profile. However, this approach looks at each track in the user profile as if they were independent with respect to the other selected tracks. This assumption likely conflicts with reality, as there is a high chance of a selected track having some sort of dependency with the other selected tracks, for example because they create a narrative flow together. Looking at the tracks in a user profile as a whole has the potential to lead to a playlist with more acoustic similarity. This aspect is beneficial as it increases both narrative flow and mood consistency, which are important qualities of a recommend sequence of items [30]. Cheng et al. [11] showed that there is consistency in human judgement of acoustic similarity and that it is important for playlist selection and recommendation *on an individual basis*. This chapter will answer **RQ3)**: To what extent can the acoustic similarity of playlists, recommended for a group, be measured and increased?

6.1. Creating a User Profile

In order to consider all tracks in a user profile combined, the similarity score from equation 4.1 can no longer be used, since it sees each track as being independent. An alternative approach is to assume that any audio feature that lies in an "acceptable" range is enjoyable for a person. This range can be created by looking at the highest and lowest audio feature value in the set of selected items, effectively creating a boundary. In fact, the Spotify Recommendation API provides a similar concept, allowing to apply a minimum and maximum value for each audio feature, excluding tracks that fall outside these limits. Since there are eight audio features provided by the Spotify API, this also creates eight boundaries that then represent a user profile. As a slight departure from the way Spotify uses audio features for recommendation is that one audio feature falling outside a boundary does not immediately exclude this track as a whole. Rather, the *boundary score* of a track is given by the amount of audio features of that track that fit in the respective boundary. The equation for calculating the boundary score can be found at equation 6.1.

$$\text{boundary\_score}(u_k, t_i) = \sum_{f \in F} \begin{cases} 1, & \text{if } \mathtt{min}(b(u_k)_f) \leq t_{i,f} \leq \mathtt{max}(b(u_k)_f) \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

where: t_i = A track
$t_{i,f}$ = The audio feature value of a track
u_k = A user
$b(u_k)$ = The boundaries calculated for the user
F = The set of all audio features

Taking Into Account the Distribution of Audio Features

However, only some audio features have a normal distribution. In fact, all but three features have a skewed distribution, some even having virtually only a single peak. This means that if a user profile happens to have a track that is in the tail of such a distribution, virtually all tracks will be considered acceptable for that feature when calculating the boundary score. Additionally, not all values within a boundary are guaranteed to be equally acceptable. As an exaggerated example, consider a person who likes only very calm or very energetic tracks, and thus has a boundary that spans most of the range of the `energy` audio feature. This person would not enjoy a track that has a more moderate energy level, yet this track would be considered acceptable for the boundary score.

A way to counter this issue is to add weights to the score, based on the distribution of the tracks within each audio feature and how well the audio feature of a track fits within the respective distributions. By breaking up the distributions into bins, this *histogram score* can be calculated by dividing the number of tracks in this bin by the number of total tracks in the user profile, as can be found in equation 6.2.

$$\text{histogram\_score}(u_k, t_i) = \sum_{f \in F} \frac{h(u_k, t_i)_f}{n_u} \tag{6.2}$$

where:	t_i	= A track
	u_k	= A user
	$h(u_k, t_i)$	= Returns the number of items in a bin
	n_u	= Total number of tracks for the user
	F	= The set of all audio features

A major caveat to the histogram score is that using the five tracks selected by each person are insufficient, as there are simply not enough data points to create a meaningful histogram. To counter this, the set of tracks that a person interacted with is available. This dataset provides the five selected tracks and a list of tracks that were considered by the person to be included in the final selection. While more noisy, it can still be used to learn more about the person and their preferences. Almost all participants searched for tracks while filling in their tracks, so there will be an overall increase of information. An added benefit of this is that the audio feature distributions of the set of interacted items seem to match the respective distributions provided by Spotify better, when compared to the set of selected items.

Looking at Audio Features Outside the Boundary

Another issue with both the histogram score and the boundary score is that feature values which fall just outside of the boundary or the histogram bin will get a score of zero. This is still the case if the values differ by a very small margin. Since the last two methods calculate a value that represents a chance that a track is acceptable based on a feature, yet another approach might be to fit a kernel density estimator (“KDE”) to each audio feature. A kernel density estimator is used to estimate probability density functions

Kernel density estimators are used to estimate the probability density function of a random variable and were created independently in the second half of the twentieth century by both Parzen and Rosenblatt [40, 52]. Once an estimator has been fit to the variable, the probability that a new value belongs to the data can be calculated. The performance of a KDE can vary widely based on the value of the bandwidth, a free parameter. It is too time consuming to find the optimal bandwidth of each estimator, one for each audio feature and person, manually. The library used for the implementation of the kernel score, `scikit-learn`[1], conveniently provides a method to find the optimal value by doing an exhaustive search over the possible values. Each estimator used later in this chapter has a bandwidth size that has been found by using this method.

Since kernel density estimation results in a continuous function, values that are outside of a boundary will also have a small, non-zero, value. Since the probability of a single point in a probability density function always returns zero, a region around the value needs to be used to calculate the resulting kernel score. The general equation can be found in equation 6.3, a region size of 0.05 and 100 steps were

[1]`https://scikit-learn.org/stable/index.html`

used for calculating the scores in this chapter.

$$\text{kernel\_score}(u_k, t_i) = \sum_{f \in F} \sum \exp\left(\text{kde}(t_{i,f} - b, ..., t_{i,f} + b)\right) * \frac{2b}{n-1} \tag{6.3}$$

where:	t_i	= A track
	$t_{i,f}$	= The audio feature value of a track
	u_k	= A user
	$kde(...)$	= Returns the kernel density estimation for set of values
	F	= The set of all audio features
	b	= Region size ($= 0.05$)
	n	= Number of steps ($= 100$)

Examples of Each Scoring Method

An example of all scores (boundary, histogram, and kernel) can be found in appendix B. Figures B.1 and B.2 show a visualisation of two survey participants and the scores, which are listed in more detail in table B.2. These users were picked from the same group, and both example tracks were selected by them during the experiment.

It should be noted that while all three scores have a theoretical range between zero and eight, this does not mean values of two different scores can be compared with each other. The boundary score will naturally have a higher score than the histogram and kernel score. In fact, only in unique cases can the histogram score reach values close to the maximum of eight. For the kernel score this is even less likely.

6.2. Results

As a first sanity check, the three scores were compared with the ratings from the survey, specifically from the ”Like” category. If the assumption of acoustically related tracks being more enjoyable is true, higher rated tracks should also have higher scores. Although the similarity score does not combine all user tracks into a profile, it was still added as an extra comparison. The average scores for each rating can be found in table 6.1, scores not fitting the expected trend are marked in italics. Both the similarity and boundary scores have values that do not fit the predicted trend, while the histogram and kernel scores do match the expected trend. The latter two also has as added benefit that it appears to follow a normal distribution, which means it can more easily be used for t-tests.

Table 6.1: Average boundary scores compared with ratings from the survey. Scores not fitting the expected trend are marked in italics, the number of tracks per rating is listed in brackets.

	Survey Rating				
Score Method	**1** (48)	**2** (126)	**3** (254)	**4** (336)	**5** (436)
Similarity	0.87	*0.86*	0.88	0.89	*0.88*
Features	6.56	6.71	7.08	*7.02*	7.46
Histogram	2.66	2.67	2.90	2.94	3.02
Kernel	1.55	1.56	1.64	1.66	1.68

Generating New Playlists

All three scores, similarity, boundary, and histogram, can be used to compare playlists for acoustic similarity. This can be done for the playlists originally used in the experiment, but also for newly generated ones. The latter will, of course, lack the accompanying survey and ratings, but the calculated scores can be used to compare the acoustic similarity of the new playlists and the original. This allows for creating playlists with different user profiles, for example by using the set of all items a person interacted with. Since it is not guaranteed that the five selected tracks are also the ones with the highest scores, this method can also provide insight on whether there were items that could have been a better choice (e.g. lead to a more enjoyable playlist) over the ones now selected. Due to the relatively simple set up of the survey, it is impossible to know why participants selected certain tracks over others. This also makes it difficult to understand why sometimes lower scoring items were chosen over other, higher scoring, items. The following data sets were compared with the originally generated playlists: the set of interacted tracks, the set of interacted tracks minus the ones that were also selected, and a new run of the selected tracks as baseline.

Comparing the Playlists and Different Scores
The results can be found in table 6.2. These were obtained by first generating new playlists in the same way as described in chapter 4, with the only difference being the different set of tracks that was considered as user profiles. Notably, the set of selected items should create a near copy of the original playlists. Because the histogram score requires the use of the interacted items as user profile to give a proper score and to keep the comparison consistent, the same set of tracks were used for the similarity and boundary scores too.

Table 6.2: Average scores and t-tests from newly generated playlists with different user profile data. The average scores from the original playlists are shown as comparison. Significant differences are marked in italics.

Track Set Used	Similarity	Boundary	Histogram	Kernel
	Probability Weighted Sum			
Original	0.91	7.06	2.89	1.63
Selected	0.92	7.12	2.98	1.65
Interacted	0.91	7.07	2.82	1.61
Interacted - Selected	*0.89*	*6.85*	*2.70*	*1.57*
	Fairness			
Original	0.91	7.11	2.93	1.64
Selected	0.91	6.96	2.92	1.62
Interacted	0.91	7.07	*2.78*	*1.59*
Interacted - Selected	0.91	7.06	*2.78*	*1.59*
	Least Misery			
Original	0.93	7.26	2.95	1.68
Selected	0.93	7.26	2.98	1.68
Interacted	*0.94*	7.36	2.94	1.69
Interacted - Selected	0.93	7.26	*2.86*	1.66

From the table it can be seen that there are in fact some differences between the original playlists and the one generated with the same dataset when using *Probability Weighted Sum* and *Fairness*. This can be explained by the fact that randomisation is involved in both of the strategies. With the *Fairness* strategy, a random user profile is selected to provide the first track. There is further randomisation as the implementation of the algorithm picks the first selected track, but the used system environment does not guarantee to keep the order of tracks consistent. The nature of *Probability Weighted Sum* is to generate random results, so a different result can be expected.

When the *Probability Weighted Sum* or *Fairness* aggregation strategies are considered, it appears that adding more items leads to less acoustic similarity. While most different were not significant enough, this can still be considered surprising. Especially as analysis on the individual track scores showed that there were often tracks that scored higher than the selected items. This means that, at least for an individual, there were often better options than were included in the playlist. This logically means that there is also a chance for a playlist to be better if those higher scoring items are included. However, the important aspect here is that such an item would only be better for the individual and not the group as a whole. The *Fairness* strategy does not take into account the preferences of the group at all and the recommended items are only based on the selected items of each person. For *Probability Weighted Sum*, adding more items has as added effect that it lowers the weights of the items across the board, which would also lower the chance of a playlist having a high acoustic similarity.

The only outlier is *Least Misery*, which has results that are similar or even slightly better compared to the original playlists. This is once again consistent with the design of the strategy. As it is the only strategy of the three that is both deterministic and explicitly takes into account the preference of the group, adding more items will only increase the chance of an acoustically similar track being available.

Potential Problems With the Scoring Methods
The somewhat surprising results of acoustic similarity in the *Probability Weighted Sum* and *Fairness* strategies also highlight a problem with methods used to calculate the scores. All three scoring methods are not invariant with respect to the number of tracks in a user profile. The similarity score takes the average of different values, which likely converges to a certain score as more items are added. In a similar fashion, one aspect of calculating the histogram score is to divide the number of items in a bin by the total number of items. This also leads to a decrease in score if more items are added. The opposite happens with the boundary score, as there is a likely increase in the boundary score as the number of items increases. This is because only extreme values are used, so adding more items gives a higher chance of the boundaries being wider.

7 Discussion

In this thesis, two experiments were shown that studied different facets of group recommendation. This chapter will interpret the results and answer the research questions stated in section 1.2. There were issues and limitations that arose during and after the experiments, which will be discussed in this chapter too. Finally, recommendations for future research will be given.

7.1. Answering the Research Questions

RQ1): How can historical data be used to improve group recommender systems in the music domain?

The purpose of the first experiment and research question was to find out if it is possible to learn the taste of a person through historical data. By comparing the tracks that were filled in by the survey participants with the historical data and two other datasets it was found that historical data is better than the baseline. Using audio features and cosine similarity as metric showed that historical data, and the playlists generated through Spotify's recommendation API performed better than randomly generated playlists. Performing Welch's t-test confirmed that these differences are significant. When comparing the random dataset with either of the other two, the p-values were below the threshold, which means the null hypothesis could be rejected. When comparing the real dataset and the one using recommended tracks, the null-hypothesis was unable to be rejected. This means that there is some overlap between these datasets in all time ranges.

At the same time, none of the time ranges appear to be good at finding the specific tracks or artist a person wants to listen to. While high outliers were present, most participants chose different tracks than they have been listening to on Spotify.

These two results combined answer **RQ1)**: Historical data can be used to *estimate* the music taste of a person to a reasonable degree, which can improve the performance of a group recommender system by speeding up the process. However, it cannot be used as a *replacement* for letting a person explicitly stating their preferences.

RQ2): To what extent does the performance of a group recommender system vary when considering different aggregation strategies?

The second experiment started as a comparison between aggregation strategies after a gap in research on different aggregation strategies in a real-life scenario was noticed. The experiment compared three different aggregation strategies: *Probability Weighted Sum*, *Fairness*, and *Least Misery*. After participants selected five tracks, they were asked to listen to and rate each playlist in three categories: enjoyment, whether they happy with the inclusion of their tracks, and suitableness.

The ratings obtained from the survey showed that using the widely used *Probability Weighted Sum* strategy results in recommendations that are slightly more enjoyed by people, but that this strategy is not convincingly better than the other two alternatives. *Probability Weighted Sum* actually scores lower than *Fairness* and *Least Misery* in the other two categories, "Selection" and "Suitable".

Participants were also able to rate each track for each category individually. As expected, the tracks a person selected themselves were, on average, rated higher than tracks they did not select.

Surprisingly, a higher rating was also observed in tracks chosen by others, that directly followed the person's own items. This effect is present for both the "Like" and "Suitable" category, but is strongest in the former. The ratings of items following a selected items in the playlists generated using *Probability Weighted Sum* or *Least Misery* were significantly different. In fact, it is absent for playlists generated by the *Fairness* strategy. It is possible that this is an example of *assimilation*, as described by [31].

Answering **RQ2)**: The performance of a group recommender in a real-life scenario remains fairly constant depending on the aggregation strategies used in this experiment. There are some small differences, but these were not significant. Two of the aggregation strategies showed evidence of assimilation, which can be regarded as an increase in performance.

RQ3): To what extent can the acoustic similarity of playlists, recommended for a group, be measured and increased?

In computing similarity scores, the group recommender in the second experiment considers each track in the user profile independently from the other user-selected items. While considering these tracks in sequence is a way to express a user profile, considering all tracks and their acoustic features as a user profile might give a better sense of the taste of the user. Three scores (boundary, histogram, and kernel) were introduced to calculate how acoustically similar a track is, given a user profile. The data of the second experiment was used to further explore this.

New playlists were generated with different sets of tracks in order to see if it is possible to increase the acoustic similarity of playlists. These new data sets used the items that participants interacted with, since this has as added benefit that it allows to explore why certain tracks were not chosen. Surprisingly, the playlists generated with the *Probability Weighted Sum* and the *Fairness* strategies and using the interacted items as dataset showed a decrease in acoustic similarity in all scores, but this decrease was not always significant. On the other hand, using *Least Misery* and the interacted items showed a slight increase, but this was not a significant increase for most scores. While calculating the scores, it was also noted that the different scoring methods are not invariant with respect to the number of items that are in a user profile, which did influence the results. Because this research question was added after the experiments, the data collected is insufficient to further examine this topic.

Given these results, the answer to **RQ3)** is inconclusive. It does appear that acoustic similarity can be measured with the introduced methods, but there is no hard evidence for this. Similarly, where an increase of acoustic similarity was expected, the data instead showed a decrease. Further research is needed to explore this.

7.2. Reflection on the Experiments

Through the feedback obtained from both experiments and while processing data, several shortcomings and limitations were found.

The First Experiment

People participating in the first experiment were asked to select tracks that they "would want to listen to". The feedback obtained from this experiment showed that this was interpreted in different ways. For example, some participants indicated that the tracks they chose represented their music taste as whole, which is not necessarily the same as what they want to listen to currently. Other feedback indicated that the selected tracks were chosen with a specific purpose in mind. While no feedback showed that the objective of the survey was confusing on its own, the results show that it perhaps should have been more focused to obtain better results.

The Second Experiment

Data from the second experiment showed that the tracks that were given as an initial selection were used infrequently. This can be for a multitude of reasons, but one reason could be that what people want to listen to while alone diverges with what they want to listen to in a group. Assuming this to be true, the conclusions of the first experiment would still hold, but the direct link to group recommendation would be missing.

There were also some issues with the second experiment, with both technical issues and set-up related issues being present. The technical issues especially might have influenced the results. As the first group, consisting of four people, was filling in their survey, an issue was noted where surveys were silently failing to be stored to the database. Logs showed that this was caused by quotation marks being present in the title of a track, which raised an exception while storing the survey. Participants were told the survey had been stored while this not actually the case. After fixing this issue, the first group was asked to fill in their surveys once more. While no obvious data loss happened due to this issue, it is possible that the results of the surveys of the first group were different compared to the first time it was filled in. A different issue where the surveys of two people in different groups failed to store, showed no errors in the log and was unable to be solved in time.

The last major technical issue occurred when the same person participated in the survey twice, but in different groups. This was explicitly allowed, as long as the groups were significantly different. However, due to an unknown issue[1], submitting tracks for the second time caused the tracks to be overwritten in the first user profile. The selected tracks were luckily able to be restored, but the tracks that were seen or interacted with were lost. This affected four people in total.

No historical data, except for the five tracks that were given as an initial selection, was collected in the second experiment. In hindsight, historical data could have been retrieved, since the consent given by participants in the second experiment also consisted of allowing data collection. This would have increased the amount of data that is available to be used in the first experiment, which only could have improved the conclusions of the experiment.

Choosing a Distance Metric

Finally, there was one issues related to the set up of the experiments that could potentially have influenced the results. Cosine similarity was selected as similarity metric for both experiments, but the substantiation of choosing this metric could have been better. The data for the first experiment is still available and several similarity metrics were compared with each other. Since they all showed similar results, cosine similarity was also used for the second experiment. In the second experiment the similarity metric was used for estimating the taste of a person and subsequently generating the recommendations. Since this step cannot easily be redone, more research should have been done to make sure that cosine similarity was a suitable choice before making the final choice.

7.3. Recommendations for the Future

Based on the results listed in this chapter, some recommendations for future research can be made. The second experiment tested group recommendation in a real-life setting, but the groups were unable to actually meet up in real life due to the COVID-19 pandemic. The second experiment could be repeated once groups are able to meet up again. This new experiment could both be used to get results that closer match reality, but also to measure the different psychological effects that should be present in group settings. This experiment could also be used to test different aggregation strategies or the same strategies adapted to be more fair. The individual track ratings in the second experiment showed an effect that could be related to *assimilation*, a new experiment can explore this effect further.

As listed in section 2.3, the context of a person listening influences the way they perceive music. This has seen some recent research in individual recommendation ([19, 34]), but has not been given much attention in group recommenders. While in the second experiment, the purpose of a group listening to music was required for groups to participate, this only served as a way to give people inspiration for selecting music. A future experiment could take into account the context of the group, and perhaps even of the individual people, to improve the performance of a group recommendation algorithm.

[1]Either due to a misunderstanding of the used database or to a bug in a beta version of a library

8 Conclusion

The main purpose of this study was to delve into some of the gaps in literature on group recommendation, specifically in the music domain. Specifically, gaps were noted in the “acquiring individual preferences” step, as research in the music domain is inconclusive on if historical data can be used to implicitly acquire the preference of an individual. Moreover, research in aggregation strategy selection is popular, but often lacks a comparison between different strategies or uses data that does not match real life. Two experiments were performed to further research both of these topics.

The first experiment asked participants to fill in songs they want to listen to at that moment and collected historical listening data. Using audio features to calculate a similarity metric, the tracks of historical data were compared to the explicitly given tracks. In order to compare these scores, two additional datasets were created. The first was generated by retrieving recommended items with the historical data as seed and the second dataset consists of randomly retrieved tracks. The results showed that the scores of the random dataset were significantly lower than the other two, while the historical data and the recommended items showed similar scores.

A group recommendation system was created for the second experiment, which allowed to test group recommendation in a real-life music group playlisting scenario. Group members were asked to explicitly provide five tracks, from which three playlists with different aggregation strategies were generated, using the same similarity scores as in the first experiment. After listening to these playlists, they were asked to rate the playlists and tracks in three questions. The ratings obtained from the survey showed that using the widely used *Probability Weighted Sum* strategy results in recommendations that are slightly more enjoyed by people, but that this strategy is not convincingly better than the other two alternatives: *Fairness* and *Least Misery*.

Thus far, the tracks in a user profile have been considered to be independent of each other. This is a mismatch with reality, as there is a high chance that those tracks have some sort of relation. Three new methods of calculating acoustic similarity were introduced to research this. These acoustic similarity scores create a user profile in which the audio features of the tracks are used to expose these dependencies. By generating new playlists with the tracks that people interacted with in the second experiment, an extra benefit was that it allowed to explore if certain tracks, which were not selected, would have increased the performance of the generated playlists. Calculating the scores for these new playlists surprisingly showed that the acoustic similarity of a playlist decreases when the number of tracks in a user profile increases, when generating playlists with the *Probability Weighted Sum* or *Fairness* strategy. The last strategy, *Least Misery*, showed tiny increases in acoustic similarity.

Further research in this topic, as well as further in-person group interaction to discuss alternative outcomes, will be left for future work. For reproducibility considerations, an anonymised version of the data of both experiments and the code can be found at [8].

Bibliography

[1] Liliana Ardissono, Anna Goy, Giovanna Petrone, Marino Segnan, and Pietro Torasso. Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied artificial intelligence*, 17(8-9):687–714, 2003.

[2] Kenneth J Arrow. A difficulty in the concept of social welfare. *Journal of political economy*, 58(4): 328–346, 1950.

[3] Solomon E Asch. Studies of independence and conformity: I. a minority of one against a unanimous majority. *Psychological monographs: General and applied*, 70(9):1, 1956.

[4] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126, 2010.

[5] Chris Beer. An overview of music streaming in 2018. *GlobalWebIndex. Available at: `https://blog.globalwebindex.com/chart-of-the-day/music-streaming-2018/` (accessed 3 June 2021)*, 2018.

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

[7] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. The music streaming sessions dataset. In *Proceedings of the 2019 Web Conference*. ACM, 2019.

[8] Aurél Bánsági. Your turn to roll: Exploring gaps in group recommendation research, dataset, August 2021. URL `https://github.com/Austaon/GroupRecommendationThesis`.

[9] Dennis L Chao, Justin Balthrop, and Stephanie Forrest. Adaptive radio: achieving consensus using negative preferences. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 120–123, 2005.

[10] Yen-Liang Chen, Li-Chen Cheng, and Ching-Nan Chuang. A group recommendation system with consideration of interactions among group members. *Expert systems with applications*, 34(3): 2082–2090, 2008.

[11] Derek Cheng, Thorsten Joachims, and Douglas Turnbull. Exploring acoustic similarity for novel music recommendation. *Currently Under Review*, 2020.

[12] Andrew Crossen, Jay Budzik, and Kristian J Hammond. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 184–185, 2002.

[13] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. " more of an art than a science": Supporting the creation of playlists and mixes. 2006.

[14] Berardina De Carolis. Adapting news and advertisements to groups. In *Pervasive Advertising*, pages 227–246. Springer, 2011.

[15] TU Delft. Template informed consent form. *TU Delft. Available at: `https://www.tudelft.nl/over-tu-delft/strategie/integriteitsbeleid/human-research-ethics/template-informed-consent-form/` (accessed 3 June 2021)*, 2020.

[16] Amra Delic, Julia Neidhardt, Thuy Ngoc Nguyen, Francesco Ricci, Laurens Rook, Hannes Werthner, and Markus Zanker. Observing group decision making processes. In *Proceedings of the 10th ACM conference on recommender systems*, pages 147–150, 2016.

[17] Alan P Fiske. The four elementary forms of sociality: framework for a unified theory of social relations. *Psychological review*, 99(4):689, 1992.

[18] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[19] Boning Gong, Mesut Kaya, and Nava Tintarev. Contextual personalized re-ranking of music recommendations through audio features. *arXiv preprint arXiv:2009.02782*, 2020.

[20] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 505–514, 2013.

[21] Elaine Hatfield, John T Cacioppo, and Richard L Rapson. Emotional contagion. *Current directions in psychological science*, 2(3):96–100, 1993.

[22] Anthony Jameson. More than the sum of its members: challenges for group recommender systems. In *Proceedings of the working conference on Advanced visual interfaces*, pages 48–54, 2004.

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

[24] Ralph H Kilman and Kenneth W Thomas. *Thomas-Kilmann Conflict Mode Instrument 1974*. Xicom, Incorporated, 1974.

[25] Vlastimil Klima. Tunnels in hash functions: Md5 collisions within a minute. *IACR Cryptol. ePrint Arch.*, 2006:105, 2006.

[26] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[27] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.

[28] Jesús Omar Álvarez Márquez and Jürgen Ziegler. Preference elicitation and negotiation in a group recommender system. In *IFIP Conference on Human-Computer Interaction*, pages 20–37. Springer, 2015.

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

[30] Judith Masthoff. Group recommender systems: aggregation, satisfaction and group attributes. In *recommender systems handbook*, pages 743–776. Springer, 2015.

[31] Judith Masthoff and Albert Gatt. In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. *User Modeling and User-Adapted Interaction*, 16 (3):281–319, 2006.

[32] Joseph F McCarthy. Pocket restaurantfinder: A situated recommender system for groups. In *Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems*, volume 8, 2002.

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

[34] Alessandro B Melchiorre and Markus Schedl. Personality correlates of music audio preferences for modelling music listeners. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, pages 313–317, 2020.

[35] Mark Mulligan. Mid-year 2018 streaming market shares. *MIDiA. Available at:* `https://www.midiaresearch.com/blog/mid-year-2018-streaming-market-shares/` *(accessed 3 June 2021)*, 2018.

[36] Shabnam Najafian and Nava Tintarev. Generating consensus explanations for group recommendations: An exploratory study. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 245–250, 2018.

[37] Thuy Ngoc Nguyen and Francesco Ricci. A chat-based group recommender system for tourism. In *Information and communication technologies in tourism 2017*, pages 17–30. Springer, 2017.

[38] Thuy Ngoc Nguyen, Francesco Ricci, Amra Delic, and Derek Bridge. Conflict resolution in group decision making: insights from a simulation study. *User Modeling and User-Adapted Interaction*, 29(5):895–941, 2019.

[39] 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.

[40] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[41] Adam Pasick. The magic that makes spotify's discover weekly playlists so damn good. *Quartz. Available at:* `http://qz.com/571007/the-magic-that-makes-spotifys-discover-weekly-playlists-so-damn-good/` *(accessed 3 June 2021)*, 2015.

[42] George Popescu. Group recommender systems as a voting problem. In *International Conference on Online Communities and Social Computing*, pages 412–421. Springer, 2013.

[43] George Popescu and Pearl Pu. What's the best music you have? designing music recommendation for group enjoyment in groupfun. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 1673–1678. Association for Computing Machinery, 2012.

[44] Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. *CoRR*, abs/1301.2303, 2013.

[45] Lara Quijano-Sanchez, Juan A Recio-Garcia, and Belen Diaz-Agudo. Happymovie: A facebook application for recommending movies to groups. In *2011 IEEE 23rd international conference on tools with artificial intelligence*, pages 239–244. IEEE, 2011.

[46] Lara Quijano-Sanchez, Juan A Recio-Garcia, Belen Diaz-Agudo, and Guillermo Jimenez-Diaz. Social factors in group recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–30, 2013.

[47] Lara Quijano-Sanchez, Christian Sauer, Juan A Recio-Garcia, and Belen Diaz-Agudo. Make it personal: a social explanation system applied to group recommendations. *Expert Systems with Applications*, 76:36–48, 2017.

[48] Juan A Recio-Garcia, Guillermo Jimenez-Diaz, Antonio A Sanchez-Ruiz, and Belen Diaz-Agudo. Personality aware recommendations to groups. In *Proceedings of the third ACM conference on Recommender systems*, pages 325–328, 2009.

[49] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3): 56–58, 1997.

[50] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[51] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 2004.

[52] Murray Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956.

[53] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[54] Thomas Schäfer, Peter Sedlmeier, Christine Städtler, and David Huron. The psychological functions of music listening. *Frontiers in psychology*, 4:511, 2013.

[55] Christophe Senot, Dimitre Kostadinov, Makram Bouzid, Jérôme Picault, Armen Aghasaryan, and Cédric Bernier. Analysis of strategies for building group profiles. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 40–51. Springer, 2010.

[56] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4): 35–43, 2001.

[57] Andreas Töscher, Michael Jahrer, and Robert M Bell. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, pages 1–52, 2009.

[58] UMBC. Use of electronic signatures for documenting informed consent. *UMBC. Available at:* `https://research.umbc.edu/use-of-electronic-signatures-for-documenting-informed-consent/` *(accessed 3 June 2021)*, 2020.

[59] B. L. Welch. The generalization of ‘student’s’ problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35, 01 1947. ISSN 0006-3444.

A Informed Consent statements

This appendix contains the full informed consent statements for the first and second experiment.

A.1. First experiment

This text was shown on the first page of the web survey.

> This study is being done by Aurél Bánsági from the TU Delft. The purpose of this research study is to investigate the kind of music a person wants to listen to. Taking part in the study will take approximately 10 minutes to complete.
>
> We believe there are no known risks associated with this research study; however, as with any online related activity the risk of a breach is always possible. To the best of our ability, your answers in this study will remain confidential and any personal data is anonymised by hashing the identifiable data.
>
> If you're interested in the result or have any questions, my email address can be found at `abansagi.nl/contact`.
>
> Your participation is entirely voluntary, and you can withdraw at any time. To start, please first log in to Spotify by pressing the button below. This is to be able to search for songs and store some, anonymised, user profile data.

A.2. Second experiment

The following text is shown when a person is creating a session.

> Here you can create a new session. However, you will first have to explicitly give consent that you want to join the experiment. Your participation is entirely voluntary, and you can withdraw at any time. If you have any questions, feel free to send an email to: A.H.J.Bansagi@student.tudelft.nl
>
> In order to start the session, please first fill in your e-mail address to ensure we can get back to you when the session is ready to proceed. You also have to give the session a purpose, something for which you and the other group members need a playlist for. For example, if you want to do a group studying session, then you can fill in "Studying" as a name. This purpose will also be used later in the closing survey to check if the playlist was generated correctly.
>
> After starting the session, you can invite people to join the session on the next page. The invited people will get an email to join the experiment. Once they have joined, there will be a checkmark next to their email address. If you are happy with the number of people who have joined, you can start the session on the same page. You can also come back to it later, in case it takes a little while for people to confirm they want to join.
>
> The experiment will consist of two stages. First you will be asked to fill in five songs that fit the theme of the group. You can use any song that is on Spotify for this. When everyone has

> filled in their choices, you will be send another email, which has a link to the survey. In this survey you will be asked to look at and listen to three playlists and answer some questions about them.
>
> In order to create the survey, you will have to log in to Spotify, this will for example allow you to search for songs. No login details are stored, but your email address and Spotify user id will be stored so that can come back to the session later (for example when the survey can be filled in). After you give consent you will prompted to log in.
>
> To give consent, please read the questions below thoroughly and answer them truthfully. You can withdraw by closing this page.

The following text is shown when a participant joins a session.

> Welcome to the survey! Before you can join the session, you will have to explicitly give consent that you want to join the experiment. Your participation is entirely voluntary, and you can withdraw at any time. If you have any questions, feel free to send an email to: A.H.J.Bansagi@student.tudelft.nl. The experiment will consist of two stages.
>
> First you will be asked to fill in five songs that fit the theme of the group (<Session name>). You can use any song that is on Spotify for this. When everyone has filled in their choices, you will be send another email, which has a link to the survey. In this survey you will be asked to look at and listen to three playlists and answer some questions about them.
>
> In order to join the survey, you will have to log in to Spotify, this will for example allow you to search for songs. No login details are stored, but your email address and Spotify user id will be stored so that you can come back to the session later (for example when the survey can be filled in). After you give consent you will prompted to log in.
>
> To give consent, please read the questions below thoroughly and answer them truthfully. If you do not want to participate, there is a withdraw button below too.

Finally, both groups get the same informed consent form with the following questions:

- I have read and understood the study information dated <date>, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.
- I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.
- I understand that taking part in the study involves storing my email address and Spotify user id. Additionally, I understand that taking part in the study involves selecting songs and filling in a survey.
- I understand that information I provide will be used for thesis research.
- I understand that personal information collected about me that can identify me, such as my email address, will not be shared beyond the study team.
- I give permission for the playlists, selected tracks and potential feedback that I provide so it can be used for future research and learning.
- By ticking the box below, you consent that you are willing to answer the questions in this survey.

B Acoustic Similarity Example

This appendix contains an example of the different scores used in chapter 6. Two participants, from the same group, of the group recommendation experiment were selected, together with a track they each selected. Table B.1 shows the audio features of the tracks and table B.2 shows the calculated scores and a breakdown of the score for each feature. Figures B.1 and B.2 show a visualisation of the scores for each audio feature.

Table B.1: Audio feature values for two example tracks used in this appendix.

	Track t_1	**Track t_2**
Acousticness	0.0218	0.620
Danceability	0.828	0.747
Energy	0.687	0.663
Instrumentalness	0.745	5.24e-5
Liveness	0.105	0.0861
Loudness	0.866	0.843
Speechiness	0.0530	0.0545
Valence	0.528	0.606

Table B.2: Tables showing example values for the boundary, histogram, and kernel scores. Track t_1 was selected by user A, track t_2 was selected by user B.

(a) Boundary scores for example user A and B and two tracks from their selected tracks.

	Boundary Score t_1		**Boundary Score t_2**	
Feature	**User A**	**User B**	**User A**	**User B**
Acousticness	1	1	0	1
Danceability	1	0	1	1
Energy	1	1	1	1
Instrumentalness	1	0	1	1
Liveness	1	1	1	1
Loudness	1	1	0	1
Speechiness	1	1	1	1
Valence	1	1	1	1
Total	8	6	6	8

(b) Histogram scores for example user A and B and two tracks from their selected tracks.

	Histogram Score t_1		**Histogram Score t_2**	
Feature	**User A**	**User B**	**User A**	**User B**
Acousticness	0.88	0.5	0.0	0.083
Danceability	0.15	0.083	0.12	0.5
Energy	0.23	0.083	0.23	0.083
Instrumentalness	0.038	0.0	0.85	1.0
Liveness	0.35	0.17	0.19	0.33
Loudness	0.27	0.5	0.27	0.5
Speechiness	0.69	0.92	0.69	0.92
Valence	0.23	0.17	0.15	0.33
Total	2.85	2.42	2.50	3.75

(c) Kernel scores for example user A and B and two tracks from their selected tracks.

	Kernel Score t_1		**Kernel Score t_2**	
Feature	**User A**	**User B**	**User A**	**User B**
Acousticness	0.34	0.17	8.6e-4	0.037
Danceability	0.11	0.20	0.15	0.29
Energy	0.17	0.16	0.15	0.15
Instrumentalness	0.02	2.05e-12	0.34	0.39
Liveness	0.22	0.21	0.21	0.20
Loudness	0.34	0.35	0.30	0.32
Speechiness	0.33	0.35	0.33	0.35
Valence	0.17	0.11	0.16	0.13
Total	1.70	1.55	1.63	1.86

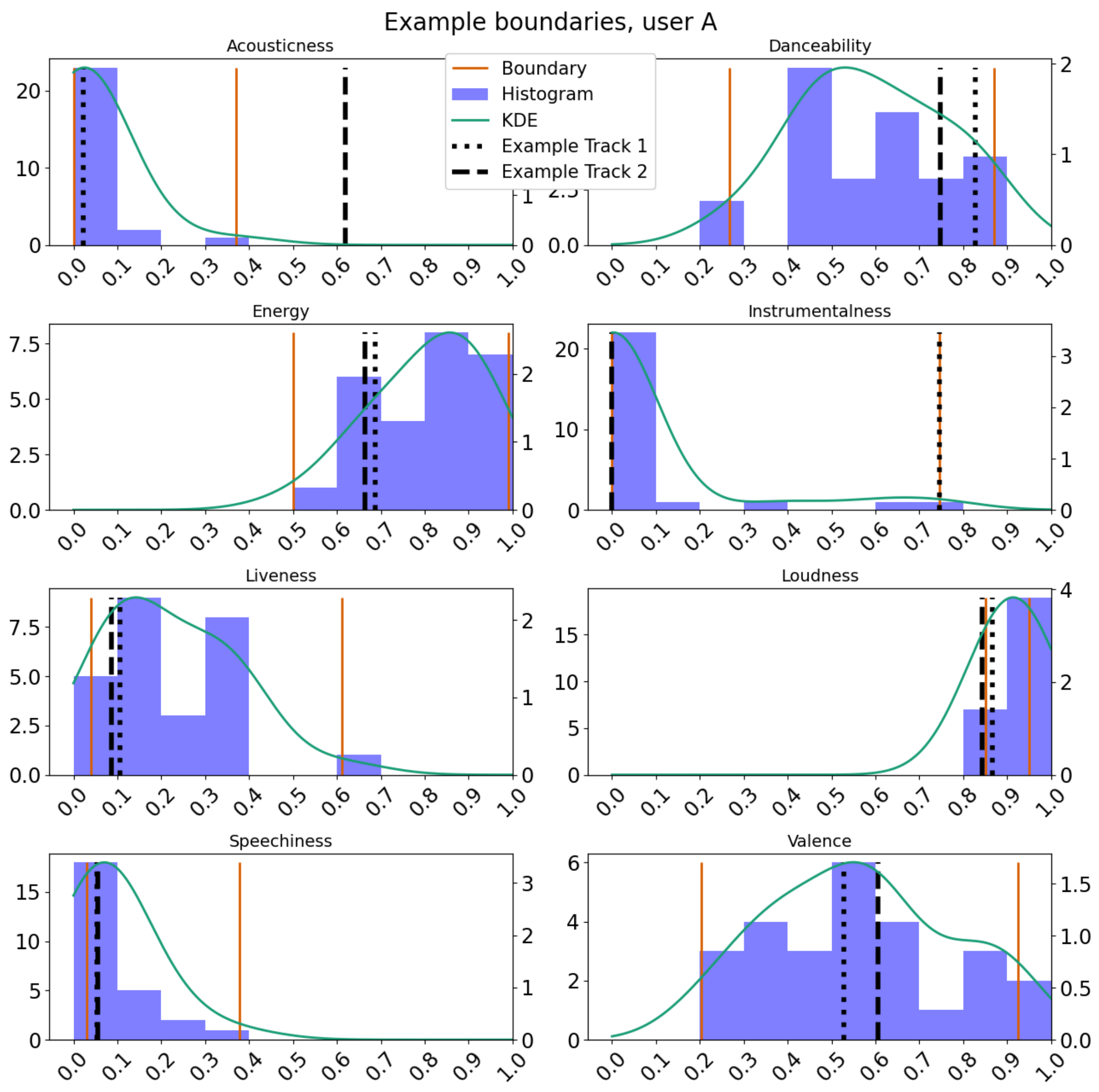


Figure B.1: The calculated boundary, histogram, and kernel scores of each audio feature for participant A. Two tracks also have their audio features plotted.

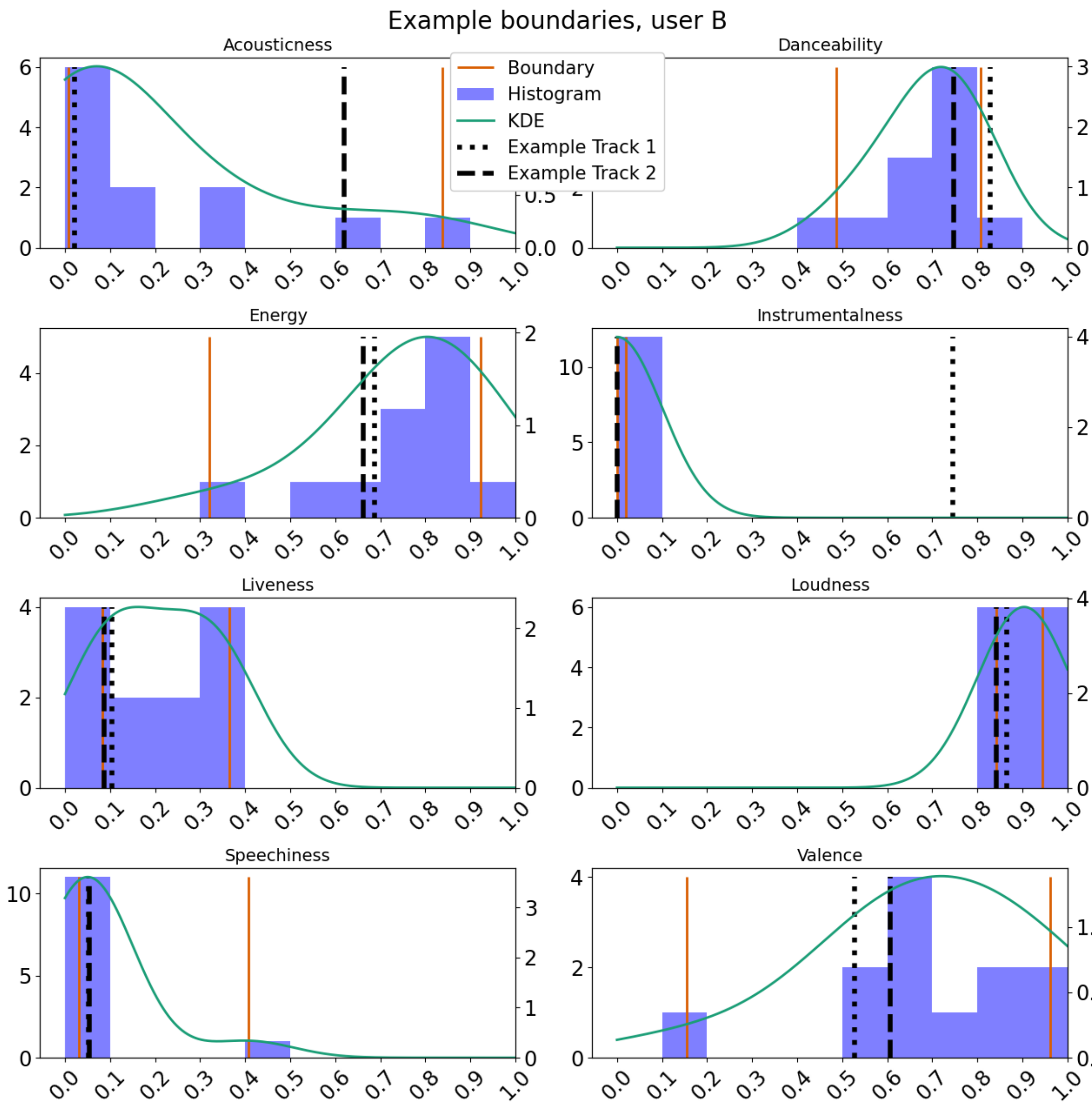


Figure B.2: The calculated boundary, histogram, and kernel scores of each audio feature for participant B. Two tracks also have their audio features plotted.

C
List of Group Purposes

This appendix contains a list of all the purposes as given by the group creators.

- Vacation
- CIG Summer BBQ
- Cozy game night[1]
- Road trip!
- The couch[2]
- Evening Snack[3]
- Girls night
- Pub Quiz Night[4]
- Wine tasting[5]
- Drinks[6]
- The troll club[7]

[1]Original name: “Gezellige Spelletjesavond”, translated from Dutch.
[2]Original name: “Sófinn”, translated from Icelandic.
[3]Original name: “Avondje snacken”, translated from Dutch.
[4]Original name: “Pub Quiz Avond”, translated from Dutch.
[5]Original name: “Wijnproeven”, translated from Dutch.
[6]Original name: “Borrel”, translated from Dutch.
[7]Original name: “De Trollenclub”, translated from Dutch.

D
Research Paper Submitted to RecSys 2021

Comparing aggregation strategies in a real-life music group recommendation scenario

ANONYMOUS AUTHORS, Anonymized, Anonymized

In group recommendation, a key question is how preferences from individuals should be aggregated into a group outcome. For comparing different possible aggregation strategies, much of the existing research in the field departs from existing preference data (e.g. ratings), but considers synthetically created groups, rather than real groups. In this paper, we describe an experiment comparing different aggregation strategies in a more ecologically valid setting. More specifically, we consider playlist creation scenarios for groups, in which participants were asked to create their own, real-life groups. While the playlists as a whole did not show any significant differences in performance, inspecting the ratings of the tracks did show some differences.

CCS Concepts: • **Information systems** → **Recommender systems**; *Rank aggregation*; *Collaborative and social computing systems and tools*; • **Applied computing** → Sound and music computing.

ACM Reference Format:
Anonymous authors. 2018. Comparing aggregation strategies in a real-life music group recommendation scenario. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With increasingly large collections becoming available for consumption and discovery, research in recommendation is currently a hot topic. Much of the research in this area focuses on recommending items for a single user, such as a movie to watch or new music to listen to. However, there also are situations, in which it is beneficial to recommend items to a group of people. For this, so-called Group Recommender Systems can be used.

Different people may have differing preferences. Therefore, for group recommenders, the key question to answer is how to aggregate the preferences of the individuals forming the group. Many different aggregation algorithms have been proposed for this [10], that each have their own strengths and weaknesses. However, there is no single best choice; even more strongly, as long as three or more items are aggregated, Arrow's impossibility theorem [3] states that there is no theoretically optimal aggregation strategy.

Domain-specific group recommender scenarios may also target different purposes with corresponding important criteria; for example, when recommending an expensive item (such as a location to visit for a holiday), it is important to recommend the single best item, while when recommending 'less expensive' items, such as songs in a playlist, having multiple 'good enough' options may suffice. Therefore, it may be interesting to consider aggregation strategies in the specific contexts of such scenarios.

Often, prior research focusing on this only chose a single aggregation strategy for an experiment, not considering alternative aggregation options. In much of the prior work that did consider multiple aggregation strategies, these strategies were performed on data with synthetic aspects: either, approaching real users for feedback, but with synthetically created, fictional group scenarios, or by synthetically generating groups from earlier user interaction feedback data (e.g. past consumptions or ratings), while that feedback is unlikely to have been occurred in the context of group scenarios. Only considering such synthetic situations can be risky, as according to [10], synthetic data can lead to inaccurate results, if the wrong metric is used.

This paper aims to empirically compare different aggregation strategies in a more ecologically valid setting: we conduct an experiment with real people, that formed real groups. For these groups, group music playlists are created, based on different aggregation strategies, but departing from items explicitly curated by the users for this purpose. In Section 2, related work will be discussed, such as pre-existing group recommenders and different voting strategies. Section 3 will discuss the group recommender experiment, followed by the conclusion.

2 RELATED WORK

2.1 General Group Recommendation

While there are many similarities between individual and group recommenders, the key difference is that the preferences of each person in the group needs to be combined. Jameson and Smyth [8] state that there are three general methods of accomplishing this: 1) Aggregating the recommendations themselves, 2) Aggregating the individual preferences, and 3) Creating a group user model. The first option is generally seen as inferior to the latter two. As for the other two, there is no clear better choice, and the suitability of a choice also depends on the domain. Examples of group recommenders using a group preference model are the GAIN System [6], INTRIGUE [2] and a group recommender for TV programs by Yu et al. [17]. While a group model is a powerful tool that can describe the preferences of a group as a whole, it is also difficult to create an accurate model. A group model will also work best if it occurs under a setting in which people can discuss the results together [8].

The other option, aggregating user preferences, is not as powerful of a tool as group modelling, but more widely researched and used. This category also falls more in line with the four sub tasks of group recommenders, also mentioned in [8]. A group recommender first needs to acquire the preferences of the group members before it can actually aggregate preferences. After it has aggregated preferences, a system needs to present the results and, optionally, help the group reach a consensus. It should be noted that in the music domain, the last two steps are not as relevant.

Regardless of the domain, the key question remains: How should preferences be aggregated? Masthoff et al. [9] provide a number of commonly used aggregation strategies, mentioning their strengths and weaknesses. For example, *Multiplicative*, *Average*, *Average without Misery*, *Most Pleasure*, and *Borda Count* strategies seem to perform well. In a later paper, Masthoff [10] notes that when people are asked to reach a decision as a group themselves (i.e. without the use of an algorithm or system), the most commonly used strategies are *Average*, *Average without Misery*, or *Least Misery*.

Numerous studies have been done to create new strategies or to find a more optimal strategy for a given domain, both with real and synthetic data. An interesting phenomenon is that simulation experiments often find the *Average* strategy to be the best (see for example Recio-Garcia et al. [15] or Senot et al. [16]), however, this is likely due to the experimental setup [9]. On the other hand, experiments with real-life scenarios often focus on one strategy and do not consider any other strategies (for example in MusicFX [11] or PolyLens [13]).

A recent development is the use of detecting personality traits, and using those to aid the decision-making. For example, the Thomas-Kilmann Conflict Mode Instrument is used to measure assertiveness and cooperativeness of people in a group [15]. This can then be used to, for example, give the ratings of a person a higher weight if their assertiveness is higher, since they are more likely to want their own items to be present. The same personality test was also used by Nguyen et al. [12] in an experiment that observed students picking a tourist location.

There are also other social effects that occur in a group, the strength of which depends on the type of relationship one has with the other people and on the personality trait. Masthoff [9] lists two such effects: emotional contagion and conformity. Emotional contagion occurs when the mood of someone else influences yours. For example, a friend enjoying a song, might cause you to like it more too. This is more likely to occur when you have a close relationship to that person. The other effect, conformity, can cause someone to change their opinions to fit in better with a group. This can be either a real change of beliefs or only expressing a different opinion. These effects are difficult to model correctly and are impossible to take into account for studies with synthetic data. This can mean that experiments performed with real people will always have, at least somewhat, different results compared to simulated experiments.

2.2 Music-specific Group Recommendation

The previous subsection was about group recommendation in the general sense. In most research on this topic, it is important to find the best item for a group, for instance what city to visit, where the cost of a wrong recommendation is quite high. However, in the music domain, it is less important to find the best item, as the cost of recommending the wrong item is substantially lower. With music often allowing for many recommendation 'shots' in the form of songs, other criteria commonly found of importance in the domain consider the inclusion of serendipity in the recommendations, and providing sequences of songs that make sense.

McCarthy and Anagnost [11] show how to include serendipity with MusicFX, a system where people are able to rate different radio stations in a fitness centre. The system works by summing all ratings together for each person present and uses the resulting scores as weights for a randomised algorithm. This way the system prevents the same radio station (i.e. the highest rated one) from playing all the time when the same people are present.

Similarly, Crossen et al. [5] also normalised ratings into a probability density function in their system, Flytrap. These ratings are acquired by keeping track of what people are listening to and then making some assumptions based on this data. That is, a song gets a high score if a person has listened to it before, or if it fits in a genre that fits with the taste of the person. They also create an automatic *DJ* that can alter the probability to make sure the created playlist still had a good taste. For example, the chance of getting two tracks by the same artist in a row is very small. This quality is important according to Masthoff [10], which mentions that the performance of group recommenders working with sequences strongly depend on the order. For the best performance, recommended sequences needs to have good narrative flow, mood consistency, and a strong ending.

This strategy of using ratings as weights in a probability density function was more thoroughly examined in a paper written by Popescu [14]. This paper calls the strategy *Probability Weighted Sum* and argues that it is easy to compute and encourages truthful answers from people. In a small scale experiment comparing different strategies, it was found that *Probability Weighted Sum* showed a tiny gain in ratings.

2.3 The Different Dimensions of a Group Recommender

As mentioned before, group recommenders can have different types of properties, such as aggregating ratings or constructing a group model. Masthoff [10] found that there are in fact six different dimensions a group recommender can be classified in: 1) Individual preferences are known or developed over time. 2) Recommendations are experienced or given as an option. 3) Active or passive group. 4) A single item or a sequence is recommended. 5) The way of obtaining the individual preferences. 6) Aggregating ratings or constructing a group model.

3 EXPERIMENT

Noticing a gap in the literature on the comparison between different possible aggregation strategies in ecologically valid group scenarios, we present a group music playlisting experiment, in which we seek to answer the following research question:

- **RQ1**: To what extent does the performance of a group recommender system vary when considering different aggregation strategies?

Before the experiment was run, its protocol, data management strategy and the forms used (e.g., informed consent) were checked for compliance and cleared by [Anonymous Institute]'s Human Research Ethics committee.

3.1 Aggregation Strategies

Since it will be too much to consider all possible aggregation strategies at the same time, our experiment focuses on three in particular: *Probability Weighted Sum*, which was extensively discussed in Section 2.2, *Fairness*, and *Least Misery*. *Fairness* and *Least Misery* are both simple to understand, but they also represent important qualities of a group recommender.

The *Fairness* strategy works, as the name implies, by letting each person pick an item, generally their highest rated, in turn. This ensures that each person has the same amount of items in the resulting list and should lead to a better performance. However, the algorithm does not take into account the tastes of the other users or the group as a whole, so there is a chance that certain items might cause more misery. The *Fairness* strategy also makes it difficult to build a strong narrative with the selected items, as each successive item must come from a different person.

The *Least Misery* strategy minimises the dissatisfaction of all members of the group by first finding the lowest rating of each item and then doing a descending sort, creating a "best of the worst" playlist. This is effective in that it recommends a set of items that all group members at least sort of enjoy. On the other hand, a common criticism of this approach is that the result lacks daring choices or outliers. This means the result might be considered enjoyable, but boring.

3.2 Experimental setup

As mentioned in Section 3.1, the strategies chosen for our experiment are: *Probability Weighted Sum*, *Fairness*, and *Least Misery*.

Our experiment consisted of several steps: a group creation step, a data collection step, and a survey evaluating generated playlists. Ideally, this last stage would have included a live in-person component; however, as the COVID-19 crisis prevented for groups to meet up in real life, and generally caused distraction in people's personal lives, we ran it in the form of an online survey instead, on which each participant could work asynchronously in their own time. Interactions were not discouraged, but at the same time not essential under this design.

For the group creation step, participants were invited to create a group with a specific self-chosen purpose in mind, for example *working out* or *studying*. They could then then invite other participants to the session using their e-mail address. Once everyone has joined the session, the person who started the group could start the next step.

In the data collection step, all group members were asked to select five tracks they would want to listen to with the group. In order to make this process easier, the five tracks they have recently listened to the most on Spotify were retrieved and presented as initial selection, but alternatively, participants had the possibility to explicitly search for songs on Spotify, a screenshot of the web survey can be found in figure 4. While participants were searching for tracks, in the background, the web page stored all the tracks that were returned from their search queries. Additionally, we also marked all tracks that the user hovered over; for later analyses, this would be a potential indicator of songs a users also more consciously considered, but did not choose for their curated list.

Once all members of the group filled their choices, the transition to the survey step would be made. The group recommender (of which further implementation details are given in 3.3) was invoked, to generate three different playlists, based on the three different aggregation strategies. Then, an automated e-mail was sent to each member of the group, to notify them that the next step was ready for their feedback.

In the survey step, all participants were asked to listen to the playlists, which were given in a random order, and to fill in a survey. This survey asked questions in three categories per playlist, as given in Table 1: whether the tracks were enjoyable ("Like" rating), whether the participants were happy with the inclusion of their own tracks ("Selection" rating), and whether the recommendations fit the purpose ("Suitable" rating). A five-point Likert scale was used to let participants answer these questions. Aside from general ratings for the entire playlists, there was also an option to rate each track individually for each category, with the middle (third) option set as default. For the general rating, no default option was set. Finally, there was also an option to leave feedback at every question and at the end of the survey, leading to 28 questions in total.

Participants were recruited on a voluntary basis via social media and by asking friends and family. All participants were based in Europe, with a focus on [anonymized country of authors]. No compensation for the participation was given, monetary or otherwise. To invite people to join a group, their e-mail addresses were used and (securely) stored before asking permission to store private data. In the invitation to join the group, an option was given to immediately delete all personal data. Before people officially join the group, they also are asked to provide explicit consent through an informed consent form.

Table 1. Specific questions asked in the survey for each category. Each of these questions was asked for every playlist, resulting in 28 questions in total.

Type		**Question Asked**
Like		
	Overall	How much did you enjoy this playlist?
	Specific	How happy are you with each of your chosen tracks?
	Feedback	Any other feedback for how much you enjoyed this playlist?
Selection		
	Overall	Are you satisfied with the position of your chosen tracks in this playlist?
	Specific	How happy are you with each of your chosen tracks?
	Feedback	Any other feedback about your chosen tracks?
Suitable		
	Overall	Do you think this playlist fits the purpose of the group session <purpose>?
	Specific	Do any of the songs fit this purpose especially well or poorly?
	Feedback	Any other feedback about how well this playlist fits the purpose?
General		Do you have any other feedback?

3.3 Group Recommender Implementation

As mentioned in section 2, a group recommender generally consists of four sub tasks, but in the music domain the first two steps are the most important. The data collection step was outlined in the section above and works by explicitly asking people to state their preferences. In this section, implementation details on the first two steps, calculating similarity scores, and generating recommendations, will be described.

3.3.1 Calculating Similarity Scores. While our experiment lets participants explicitly curate songs they personally deem suitable for the playlist, we initially do not know how participants would rate the songs curated by other participants. Considering our asynchronous working setup, as well as cognitive user load, we also could not easily have participants explicitly rating all items curated by everyone, before creating the playlists. Therefore, we approximate the rating for non-curated items by calculating acoustic similarity scores. [4] showed that there is a connection between acoustic similarity and recommendation and that acoustic similarity is consistent across different people. Their research used audio features as obtained through the Spotify API, which are the features we use in our experiment, too.

In order for the system to calculate a similarity score, it first needs to know about a person's own taste. To accomplish this, the five tracks explicitly chosen by a user are considered to represent that user's taste profile, and are given the highest score of 1.

In order to then obtain the similarity score to the user's profile for a given music track, we choose a similarity metric, in our case cosine similarity, and average the track's similarity to each of the curated user profile tracks, as expressed in Equation 1. For each of the tracks explicitly selected by other members of a group, we calculate the similarity score to a given user's profile. Based on these scores, all these other tracks will be ranked for this given user.

$$\text{similarity\_score}(u_k, t_i) = \frac{\sum_{t_j}^{u_k} \text{similarity}(t_i, t_j)}{n} \tag{1}$$

where: u_k = A given user

t_i = A given track

n = Number of tracks in a user profile

3.3.2 Implementation of Aggregation Strategies. For the implementation for *Probability Weighted Sum*, the variant described by MusicFX [11] is used. This method first squares and sums the similarity scores of a track together. The similarity score for each track is then normalised so that the scores of all tracks together sum up to one. Equation 2 also shows how the probabilities are calculated. Using the calculated values as weights, the algorithm then randomly picks ten items.

$$p(t_i) = \frac{\sum_{u_i}^{U} \text{similarity\_score}(u_i, t_i)^2}{\sum_{j}^{T} \sum_{u_k}^{U} \text{similarity\_score}(u_k, t_j)^2} \tag{2}$$

where: u_k = A given user

t_i = A given track

T = The set of all tracks

U = The set of users

The *Fairness* strategy randomly selects a group member and picks their highest rated item and removes it from their available items. In this case, this item will always be one of the items the person selected. The system then gives the next user a turn by picking their top item. This process continues, looping back to the first user if needed until ten items are selected.

Finally, *Least Misery* works by first applying a `min` function over the similarity scores for each track, which results in a list of lowest ratings for each items. These ratings is then sorted in descending order, as seen in equation 3. This results in a list with the highest “worst” similarity score being selected first. Once again, the top-ten items of this list are selected for the playlist.

$$\text{least\_misery}(t_1, ..., t_n) = sort(min(t_1, ..., t_n)) \quad (3)$$

Regardless of the aggregation strategy, the algorithm returns a list of ten recommended items. The group recommender generates one playlist per aggregation strategy, and presents the three resulting playlists to the members of the group.

3.4 Results

During the run of the experiment, 40 people participated in the survey, spread over 11 sessions. An additional three people were unable to complete the survey, but did select their tracks, which means their tracks were used for the recommendations. A total of 210 unique tracks were chosen by the 43 participants. They interacted with (i.e., hovered over and/or selected) 805 unique tracks, and 2162 unique tracks were retrieved as part of the search queries. Each person had an average of 2.6 songs per playlist; only in rare cases, a person did not have any tracks at all in a playlist. Regardless of the used aggregation strategy, this average holds, but the variance is larger for *Probability Weighted Sum* and *Least Misery* than for *Fairness*. It should be noted that the design of the first two strategies does not ensure that all group members had at least one of their own tracks in the playlist. On average, participants saw 68 different items while selecting their tracks and interacted with 25. There were no signs of survey fatigue, such as ratings later in the survey being lower.

Figure 1 shows the box plots of the positions of when participants on average found their tracks, relative to the full sequence of items they interacted with. This data shows that most participants settled on their tracks in the second half of searching for items. The box plot shows that the initial pre-filled selection of tracks that the person recently has listened to most, was used by roughly half of the participants. A total of twenty used at least one item from the historical data, while four used all five and did not search for other tracks. This shows that using historical data as initial selection was likely useful to serve as a starting point. The same figure also shows the average session length in both the total number of items seen while selecting items and the number of items that were interacted with.

The average survey ratings per playlist and per survey category can be found in Table 2. *Probability Weighted Sum* scored slightly higher in the “Like” category compared to the other two strategies, but the ratings also have a larger standard deviation. For the other two categories, *Fairness* scored higher than *Probability Weighted Sum* and *Least Misery*. Due to common feedback that the “Selection” questions was confusing, especially when tracks were not in the playlist at all, it will be excluded from further analysis.

Figure 2 shows box plots for both the “Like” and “Suitable” categories and each aggregation strategies. This figure confirms that the results for the *Fairness* strategy vary less compared to the other two. Since this strategy also has a more consistent number of tracks per person that end up in a playlist, there likely is a correlation between tracks being selected by a person, and higher ratings.

This correlation can be further examined by using the ratings given to individual tracks. While giving a 5-point explicit Likert scale rating to all individual tracks was optional (a rating of 3 was pre-filled by default), most participants

Table 2. Means and standard deviations of the survey ratings given to the playlists for each question.

	Like		**Selection**		**Suitable**	
Aggregation Strategy	**M**	**SD**	**M**	**SD**	**M**	**SD**
Probability Weighted Sum	3.98	0.86	3.55	1.11	3.88	1.09
Fairness	3.92	0.76	3.90	0.90	3.98	0.80
Least Misery	3.85	0.77	3.67	1.14	3.92	0.83

did fill in individual ratings for the tracks. 82% and and 75% of the individual ratings in the "Like" and "Suitable" category respectively had a rating that was changed from the default. A rating that was not changed can either mean the person did not want to fill in a rating, or that the default was an acceptable rating for that track.

The individual ratings can be divided in two groups: those given to tracks that were selected by the person answering the survey (referred to as "selected items"), and to tracks that were selected by others in the group. Comparing the average ratings for these groups shows that the ratings of the selected items are 0.94 higher in the *Like* category and 0.56 higher in the *Suitable* category. This is an expected result and consistent with observations from [7]. More surprising is that tracks directly following the items chosen by a person are also rated higher than expected. This effect only occurs in the *Like* category and not in the *Suitable* category. This effect is likely an example of *assimilation* [10]. Figure 3 shows the differences between the selected and not-selected items and between items following a selected item and the others[1]. Using a t-test shows that this difference is significant ($t(238) = 2.397, p = 0.017$) when looking at all ratings. While it would make the most sense for this effect to happen in playlists using *Fairness*, it is actually not present at all with the *Fairness* aggregation strategy.

4 CONCLUSION

We presented an experiment, comparing three different aggregation strategies in a real-life music group playlisting scenario: *Probability Weighted Sum*, *Fairness*, and *Least Misery*. Participating groups were asked to explicitly select five tracks each, from which the playlists were generated. They were then asked to rate each playlist in three categories: enjoyment, whether they happy with the inclusion of their tracks, and suitableness. The "Selection" category, however, was not well understood and was excluded from most results.

The ratings obtained from the survey showed that using the widely used *Probability Weighted Sum* strategy results in recommendations that are slightly more enjoyed by people, but that this strategy is not convincingly better than the other two alternatives. *Probability Weighted Sum* actually scores lower than *Fairness* and *Least Misery* in the other two categories, "Selection" and "Suitable".

Exploring the ratings given in the survey to individual tracks showed an interesting effect. While it is expected that people rate their own items higher than average, this also seems to occur in tracks chosen by others, that directly followed their own items. This effect is observed for both the "Like" and "Suitable" category, but is strongest in the former. These ratings of items following a selected items in the playlists generated using *Probability Weighted Sum* or *Least Misery* were significantly different.

In computing similarity scores, the group recommender in this experiment considers each track in the user profile independently from the other user-selected items. While considering these tracks in sequence is a way to express a user profile, more distributionally oriented approaches, that consider commonly shared acoustic properties or feature distributions of the chosen (or even interacted) tracks, may give a better sense of the user profile. We leave

[1]To prevent bias, user-selected items that directly follow another user-selected item are left out of our analysis.

such approaches, as well as further in-person group interaction to discuss alternative outcomes, to future work. For reproducibility considerations, we currently release an anonymised review version of our survey data at [1]. This data, and the underlying experimental and analysis code, will be released in formal repositories upon acceptance of this work.

REFERENCES

[1] Anonymous. 2021. *Comparing aggregation strategies in a real-life music group recommendation scenario.* https://doi.org/10.5281/zenodo.4738590

[2] Liliana Ardissono, Anna Goy, Giovanna Petrone, Marino Segnan, and Pietro Torasso. 2003. Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied artificial intelligence* 17, 8-9 (2003), 687–714.

[3] Kenneth J Arrow. 1950. A difficulty in the concept of social welfare. *Journal of political economy* 58, 4 (1950), 328–346.

[4] Derek Cheng, Thorsten Joachims, and Douglas Turnbull. 2020. Exploring acoustic similarity for novel music recommendation. *Currently Under Review* (2020).

[5] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. 2002. Flytrap: Intelligent Group Music Recommendation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces* (San Francisco, California, USA) *(IUI '02)*. Association for Computing Machinery, New York, NY, USA, 184–185. https://doi.org/10.1145/502716.502748

[6] Berardina De Carolis. 2011. Adapting News and Advertisements to Groups. In *Pervasive Advertising*, Jörg Müller and Daniel Alt, Florianand Michelis (Eds.). Springer London, London, 227–246.

[7] Amra Delic, Julia Neidhardt, Thuy Ngoc Nguyen, Francesco Ricci, Laurens Rook, Hannes Werthner, and Markus Zanker. 2016. Observing Group Decision Making Processes. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 147–150. https://doi.org/10.1145/2959100.2959168

[8] Anthony Jameson and Barry Smyth. 2007. Recommendation to Groups. In *The Adaptive Web: Methods and Strategies of Web Personalization*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 596–627.

[9] Judith Masthoff. 2011. Group Recommender Systems: Combining Individual Models. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, Boston, MA, 677–702.

[10] Judith Masthoff. 2015. Group Recommender Systems: Aggregation, Satisfaction and Group Attributes. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). Springer US, Boston, MA, 743–776.

[11] Joseph F. McCarthy and Theodore D. Anagnost. 1998. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work* (Seattle, Washington, USA) *(CSCW '98)*. Association for Computing Machinery, New York, NY, USA, 363–372.

[12] Thuy Ngoc Nguyen, Francesco Ricci, Amra Delic, and Derek Bridge. 2019. Conflict resolution in group decision making: insights from a simulation study. *User Modeling and User-Adapted Interaction* 29, 5 (2019), 895–941.

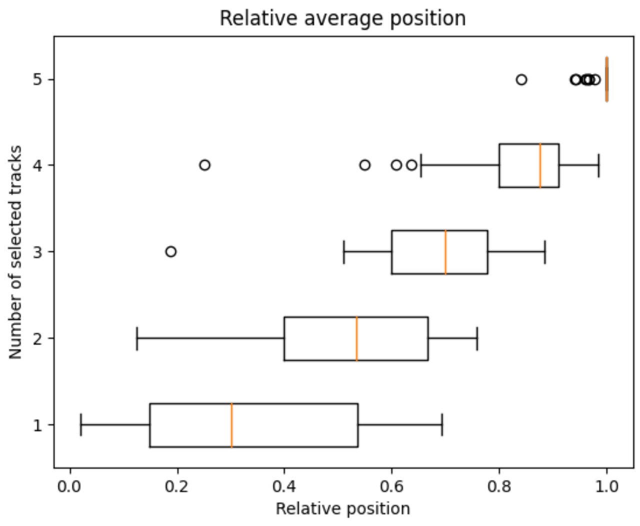
[13] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. 2001. PolyLens: A Recommender System for Groups of Users. In *ECSCW 2001: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work 16–20 September 2001, Bonn, Germany*, Wolfgang Prinz, Matthias Jarke, Yvonne Rogers, Kjeld Schmidt, and Volker Wulf (Eds.). Springer Netherlands, Dordrecht, 199–218.

[14] George Popescu. 2013. Group Recommender Systems as a Voting Problem. In *Online Communities and Social Computing*, A. Ant Ozok and Panayiotis Zaphiris (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 412–421.

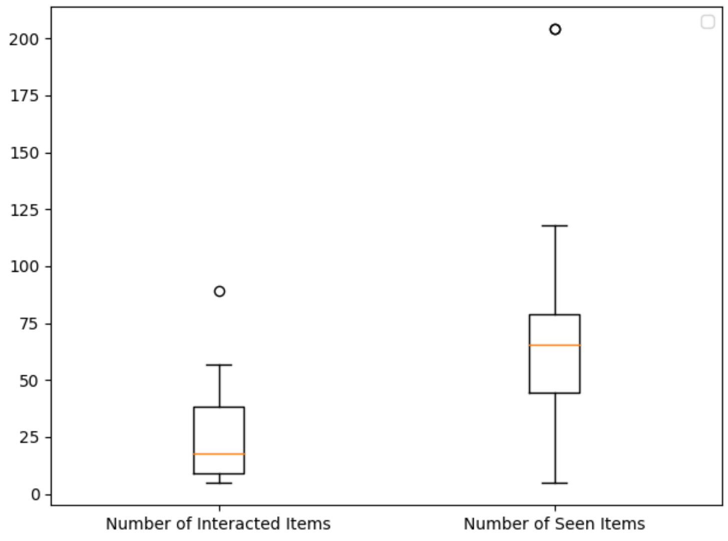
[15] Juan A. Recio-Garcia, Guillermo Jimenez-Diaz, Antonio A. Sanchez-Ruiz, and Belen Diaz-Agudo. 2009. Personality Aware Recommendations to Groups. In *Proceedings of the Third ACM Conference on Recommender Systems* (New York, New York, USA) *(RecSys '09)*. Association for Computing Machinery, New York, NY, USA, 325–328. https://doi.org/10.1145/1639714.1639779

[16] Christophe Senot, Dimitre Kostadinov, Makram Bouzid, Jérôme Picault, Armen Aghasaryan, and Cédric Bernier. 2010. Analysis of Strategies for Building Group Profiles. In *User Modeling, Adaptation, and Personalization*, Paul De Bra, Alfred Kobsa, and David Chin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 40–51.

[17] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. 2006. TV program recommendation for multiple viewers based on user profile merging. *User modeling and user-adapted interaction* 16, 1 (2006), 63–82.

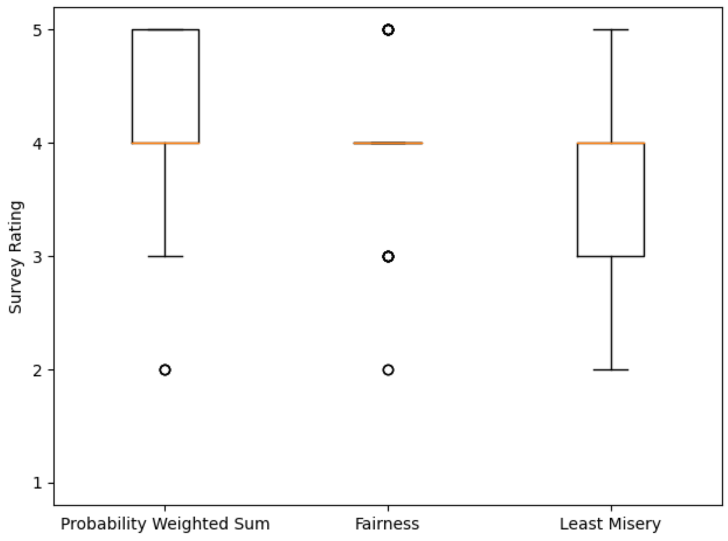


(a) Relative positions of when tracks were selected

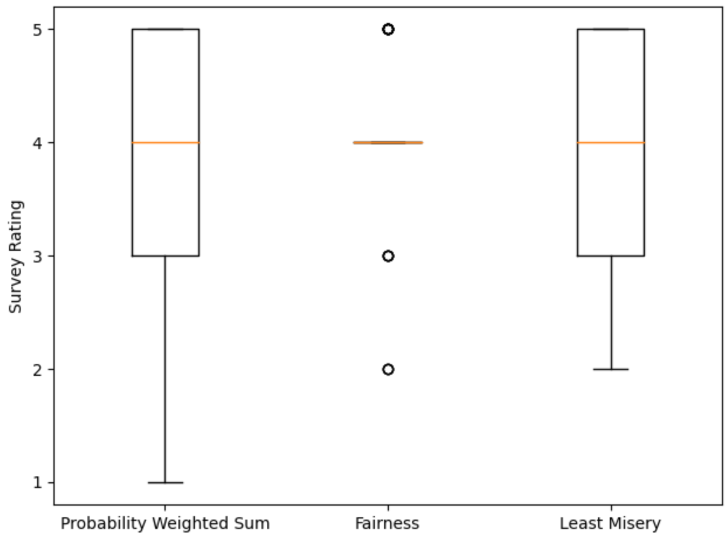


(b) Average Session Length

Fig. 1. Box plots of the (relative) positions of when the tracks were selected in the data collection step and of the average number of interacted and seen items.

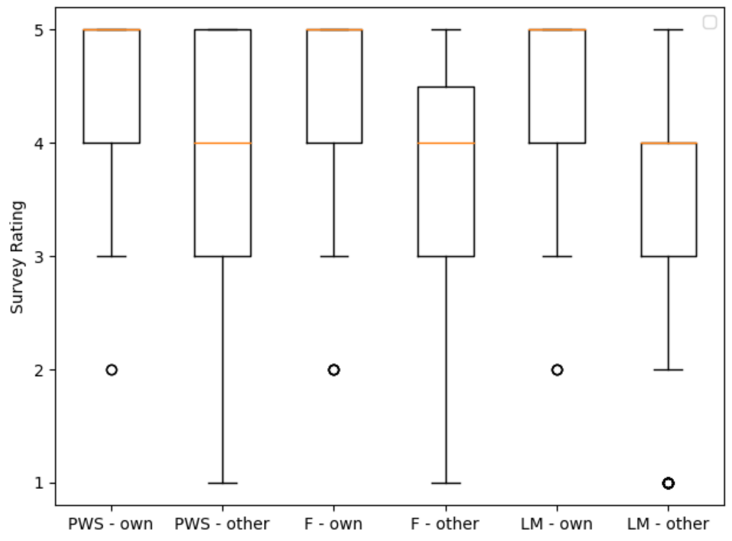


(a) "Like" category

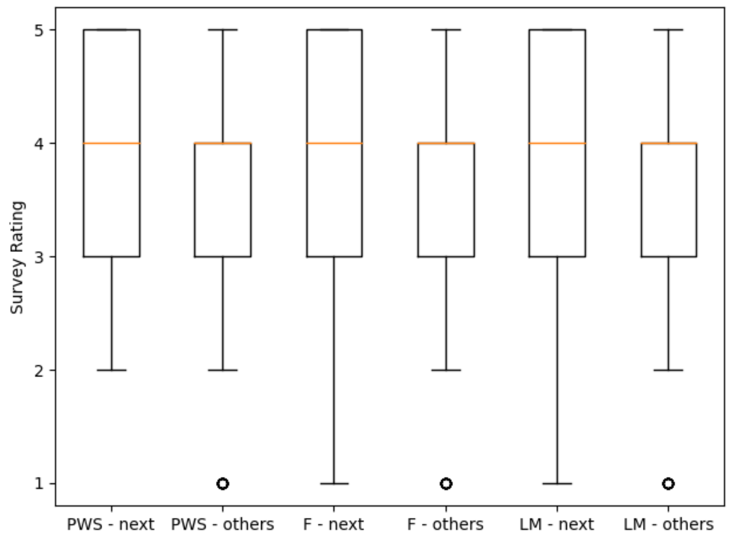


(b) "Suitable" category

Fig. 2. Box plots of the ratings obtained from the survey. Both the "Like" and "Suitable" categories are shown.

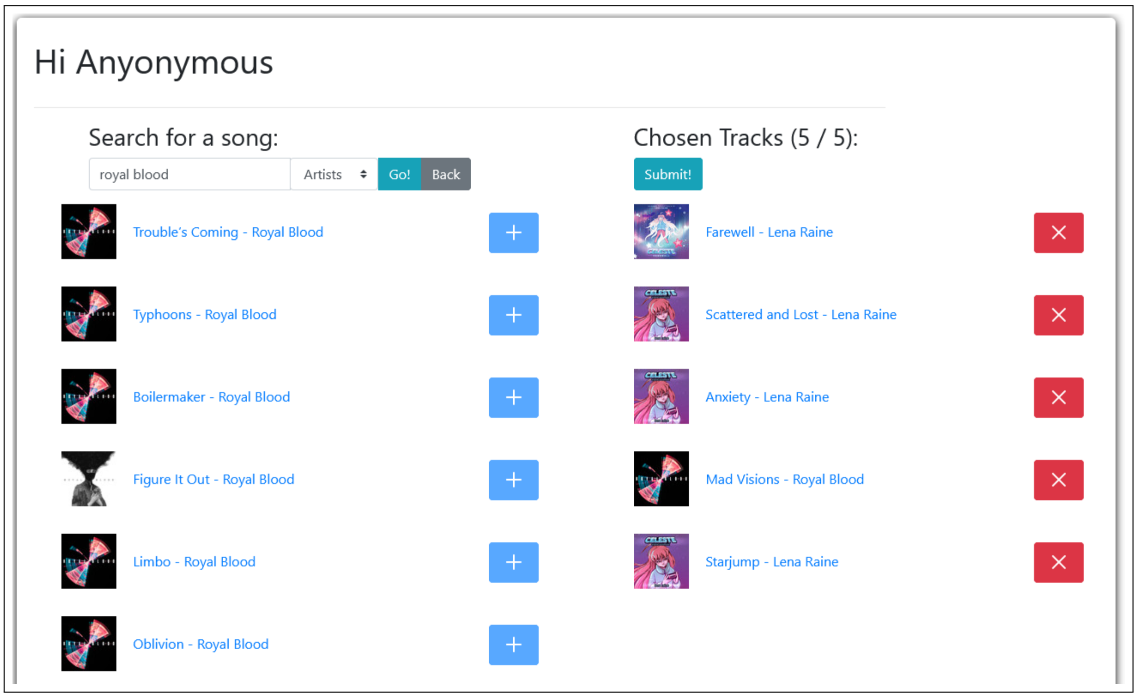


(a) Selected items vs. non-selected items



(b) Non-selected items following a selected item vs. other non-selected items

Fig. 3. Box plots of the ratings obtained from the survey divided in two ways. The first divides in selected and non-selected items. The second divides on items directly following a selected item and the other non-selected items.


Hi Anyonymous

Search for a song:

royal blood | Artists | Go! | Back

Trouble's Coming - Royal Blood

Typhoons - Royal Blood

Boilermaker - Royal Blood

Figure It Out - Royal Blood

Limbo - Royal Blood

Oblivion - Royal Blood

Chosen Tracks (5 / 5):

Submit!

Farewell - Lena Raine

Scattered and Lost - Lena Raine

Anxiety - Lena Raine

Mad Visions - Royal Blood

Starjump - Lena Raine

Fig. 4. Screenshot of the web survey. The left column is used to search for tracks, artists, or albums. The right column shows the current selected items and has a button to submit the tracks.