SQUIRREL: A Framework for Sequential Group Recommendations through Reinforcement Learning

Maria Stratigi[a], Evaggelia Pitoura[b], Kostas Stefanidis[a]

[a] *Tampere University, Tampere, Finland*
[b] *University of Ioannina, Ioannina, Greece*

Abstract

Nowadays, sequential recommendations are becoming more prevalent. A user expects the system to remember past interactions and not conduct each recommendation round as a stand-alone process. Additionally, group recommendation systems are more prominent since more and more people are able to form groups for activities. Subsequently, the data that a group recommendation system needs to consider becomes more complicated – historical data and feedback for each user, the items recommended and ultimately selected to and by the group, etc. This makes the selection of a group recommendation algorithm to be even more complex. In this work, we propose the SQUIRREL framework – SeQUentIal Recommendations with ReinforcEment Learning, a model that relies on reinforcement learning techniques to select the most appropriate group recommendation algorithm based on the current state of the group. At each round of recommendations, we calculate the satisfaction of each group member, how relevant each item in the group recommendation list is for each user, and based on this the model selects an action, that is, a recommendation algorithm out of a predefined set that will produce the maximum reward. We present a sample of methods that can be used; however, the model is able to be further configured with additional actions, different definitions of rewards or states. We perform experiments on three real world datasets, 20M MovieLens, GoodReads and Amazon, and show that SQUIRREL is able to outperform all the individual recommendation methods used in the action set, by correctly identifying the recommendation algorithm that maximizes the reward function utilized.

Email addresses: maria.stratigi@tuni.fi (Maria Stratigi), pitoura@cs.uoi.gr (Evaggelia Pitoura), konstantinos.stefanidis@tuni.fi (Kostas Stefanidis)

Keywords: Multi-round recommendations, Group recommendations, Rank aggregations
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

Recommendation systems have developed into one of the most influential part of research for projects, spanning from e-health to movie recommendations. This is because recommendation systems are to a great degree responsible for the end-users satisfaction with an application. They analyze a user's past interactions (in the various forms these interactions may take like reviews, like/dislike, clicks, etc.) to enhance the experience of the user. To achieve this, the recommendation systems have to consider not only the user's current information but also their past interactions with the system. That is, the system should not only rely on the current session but also consider historical ones and the response that the user had on them, i.e., which items the user ultimately picked or liked. The system should consider a historical *sequence of interactions* or *rounds of recommendations* for a user, instead of focusing on just the current session. This is a relatively new approach to recommendations, since typically most systems examine the user's available ratings but do not consider how well the user received the past recommended items.

In this work, we consider a sequential recommendation system as a system that considers a sequence of rounds of recommendations for users. A previous work done on sequential recommendations is [1], where user and item dynamics are captured by historical sequences on the user side and the item side, respectively. In [2], a neural network is used, and every round of recommendations is represented by an embedding of the users' preferences. [3] uses variational autoencoders (VAEs) to propose a multi-round recommendation system, which involves introducing randomness into the regular operation of VAEs in order to promote fairness, while [4] penalizes items for their historical popularity in an effort to minimize bias and promote diversity.

In addition to sequential recommendation, in recent years, greater attention is given to group recommendations, where the recommendation system needs to balance the interests of a group of users instead of focusing on just one user. For example, assume a group of friends that wants to watch a

movie. The group recommendation process is more complicated than a single recommendation system, since each group member has their own likes and dislikes and the group recommendation system needs to balance them, in order to propose a group recommendation list that ideally is relevant to all members interests.

There are many different ways to achieve group recommendations, but one of the most popular is to employ a single recommendation system for each group member and receive their corresponding recommendation lists. Then, the group recommendation system takes over and tries to combine these lists into one group recommendation list via an aggregation method [5]. Many strategies, with different advantages and disadvantages, can be applied during this aggregation step. However, even after years of research, and according to the Arrow's Theorem [6], no aggregation function has been proven to be the most effective one.

Both of these areas of recommendation systems, sequential and group recommendations, are complex on their own right, but they become even more complex when they are combined. This new area of recommendations, the *sequential group recommendations*, is important to explore in more depth. Similarly to single recommendation systems, a sequential group recommender will be able to provide to a group a more robust experience if it analyzes its past interactions with that group. Specifically, a sequential group recommender has to resolve the challenges of group and sequential recommendations simultaneously. That is, it has to balance the interest of all group members and propose relevant items to all, while accurately determining how to infer users' dynamic preferences between rounds of recommendation.

Although there are many ways one can approach group recommendations, it is not always clear which approach is the best for each test case. The vast amount of group recommendation methods and their widely distinct approaches to recommendations, make it very difficult and time consuming to not only evaluate all of them but to also see which one works best for each test scenario. Some of the methods work very well in a specific recommendation domain, for example movie recommendations, but straggle when they are applied to a different one [7]. As a result it is not an easy task to simply transfer a recommender system to another domain.

In this work, we propose a model that is able to be applied to any given domain without major alterations and minimum time invested, while simultaneously, it is also able to maintain high performance. The SQUIRREL framework, **S**e**QU**ent**I**al Group **R**ecommendations through **R**einforc**E**ment

Learning, is a model that is based on Reinforcement Learning (RL), a natural choice, since the sequential nature of RL directly mirrors the sequential nature of our problem. It consists of three main components: state, actions and reward. The state describes the current status of the group, the actions are the different group recommendation methods that can be applied and finally, the reward is the primary goal that the system wants to achieve. In most recommendation systems that goal is the group members' satisfaction with the proposed items, meaning how relevant are the items in the group recommendation list for each group member.

At each round of recommendations, the system examines the state of the group, for example, how satisfied each group member is, and based on that, it selects an appropriate action to make, i.e., it selects a group recommendation method to apply. Based on this selection, a group recommendation list is produced and returned to the group and a reward is calculated. Then, the state of the group is updated to reflect the changes made by the action, i.e., how satisfied are the group members after the latest recommendation round. Note that to simulate time passing between each round of recommendations, we augment the system with additional data (i.e., the users rate more items, hence the user profiles change between rounds of recommendations).

All the components of the model can be configured to the specifications of the user and the criteria of the test scenario, and are easily altered. For instance, if we want to evaluate a new group recommendation method, then that method is simply added as a new action in the model. If the goal of the recommender needs to change, for example, the system's new primary goal is changed from promoting satisfaction to promoting diversity, then a new reward can be defined. If the system is transferred to a new recommendation domain then a new state can be specified. However, it is worth noting that if a component changes, then the model will have to be trained again.

Since the variations of scenarios and group recommendation methods that can be applied are numerous, in this work we mainly focus on rank aggregation group recommendation methods. As the state of our model, we use the notion of user's satisfaction, i.e., how relevant the items in the group recommendation list are to each group member. Finally, we study two reward functions: one based on satisfaction and the second based on the combination between user's satisfaction and user's disagreement, which is defined as the difference between the most and least satisfied group member.

Overall, the main contributions of our work are the following:

- We introduce the SQUIRREL framework, a model that based on principles of reinforcement learning is able to identify the best possible group recommendation method to apply based on the information it has available and the goal it wants to fulfill.
- We focus on rank aggregation group recommendation methods, and exploit the notion of user satisfaction and user disagreement to define the various components of our model.
- We evaluate the SQUIRREL model and all individual methods used as actions, using three real world datasets, 20M MovieLens, GoodReads and Amazon. Additionally, we examined the behaviour of all methods on different types of groups.

The rest of the paper is structured as follows. In Section 2, we formally present the SQUIRREL model, while in Section 3, we go into more detail about all the components of the model; state, actions and rewards. In Section 4, we showcase the datasets we utilized, and in Section 5, we present the experimental process and analyze the results. Furthermore, in Section 6, we describe the related work, and finally, in Section 7, we conclude this work.

2. Reinforcement Learning for Group Recommendations

Let I denote a set of data items and U a set of users. G denotes a group of users where $G \subseteq U$. For each user u in the group, we denote as B_u^j an ordered list of recommended items, as they have been generated by a single recommender system at a specific recommendation round j for user u. At round j, the SQUIRREL model chooses an appropriate aggregation function based on the current state of the group, to combine the individual members' recommendation lists B_u^j into one group recommendation list GL_G^j.

We apply the SQUIRREL model for a sequence of rounds in order to produce μ group recommendation lists for a group G, defined as $\mathcal{GR} = (GL_G^1, GL_G^2, ..., GL_G^\mu)$. At each round, we calculate each group members' individual *user utility* scores, as well as the *group utility* scores. The first set of scores portrait how satisfied each member is with the proposed group recommendation list, as well as the disagreement between the group members, i.e., the variance between the satisfaction scores of the group members. The latter set of utility scores describe the degree of satisfaction and disagreement for the entire group. Namely, how can we exploit the user utility scores to

Notation	Definition	Notation	Definition
I	Set of items	GL_G^j	Group recommendation list at round j
U	Set of users	$\mathcal{GR}$	Sequence of group recommendations
G	Group	μ	Number of rounds in the sequence
u	User	SQUIRREL Model	
d	Item	S	State of the environment
j	Round	A	Set of actions
B_u^j	Recommendations list for user u at round j	$P_a(s, s')$	Probability of transitioning from state s to state s'
$B_{u,k}^j$	Top k recommendations for user u at round j	$R_a(s, s')$	Reward from transitioning from s to s' under action a
$p_j(u, d)$	User u's relevance score for item d at round j	π	Policy of the model

Table 1: Summary of the notations used in this work.

reflect the entire group. We further expand on these notions in Subsection 3.2.

The SQUIRREL model can be described as a Markov decision process, where an agent interacts with an environment E, in order to maximize the accumulative reward after each recommendation round. The Markov decision process can be described by a tuple of (S, A, P, R), where:

- $\boldsymbol{S}$ is a continuous space that describes the environmental state, i.e., the group state and we express it via the utility scores of the group members. We keep an individual state for each group member u at each round j. It can be defined as $s_u^j = satO(u, j)$, where $satO$ is the overall satisfaction of user u at round j. This satisfaction score will be discussed in detail in Subsection 3.1. Briefly, it describes how relevant are the data items in the group recommendation list, compared to the best case scenario for the user, which is their individual recommendations B_u^j.

- A is a set of distinct actions that consists of the different aggregation functions employed by the SQUIRREL model, where $|\mathrm{A}| = m$. These functions can range from simplistic ones, like a classic Average, to far more complex ones like SDAA first proposed in [8], without any restriction in regards to the number of actions we can include. We go in more detail about each aggregation method used in Subsection 3.3.

- $P_a(s, s')$ defines the probability to transition from state s to state s' during round j under the action a. Formally, $P_a(s, s') = Pr(s_{j+1} = s' | s_j = s, a_j = a)$.

- $R_a(s, s')$ is the reward gained from transitioning from state s to state s'. The reward describes the quality of recommendations given by the

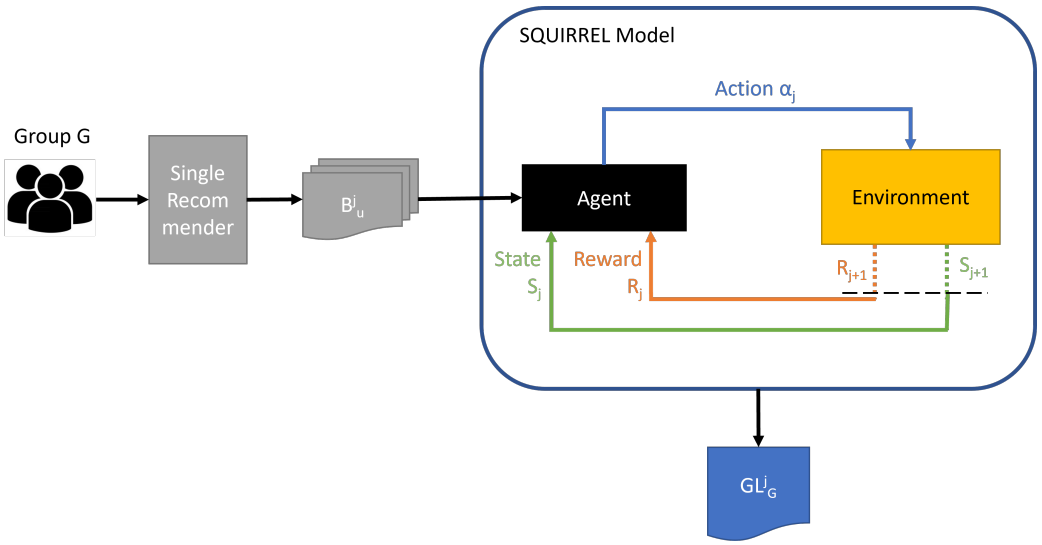


Figure 1: The SQUIRREL Model.

model. We define two reward functions by utilizing group utility scores. First, we examine only the overall satisfaction of the group by averaging the individual satisfaction of all group members. Second, in addition to the overall satisfaction, we also consider the disagreement between the group members. This is defined as the difference in the satisfaction scores between the most and least satisfied group member. We expand on these reward functions further in Subsection 3.2.

The goal of the model is to find a policy $\pi(a|s)$ that takes action $a \in A$ during state $s \in S$ in order to maximize the expected discounted cumulative reward after μ recommendation rounds:

$$max\, \mathbb{E}[R(\mu)] \tag{1}$$

where

$$R(\mu) = \sum_{t=0}^{\mu} \gamma R_{\alpha}(s, s') \tag{2}$$

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

It is worth noting that the individual component of the SQUIRREL model can be altered and fine-tuned for a specific purpose. For example, an application that wants to minimize the differences between users' perceived overall

performance may need to define a different state and/or reward, while an application that wants to find the best variable for an aggregation function may want to define a different action space. For instance, in the case of a simple Weight Sum aggregation function [9], the action space will be the different weights.

As any RL model, SQUIRREL requires a training phase. To achieve this, the system depends upon having a large amount of data to utilize for this training phase. During training, the model will learn what action (i.e., what aggregation method) is more effective to use based on the current state of the environment. If any of the model's components (state, actions or reward) is changed, then the model will have to be re-trained.

In Figure 1, we show how a recommendation round is structured in the SQUIRREL model. At the beginning of the round j, the group is given to a single user recommender system that produces a set of recommendation lists for each group member, B_u^j. These lists are then given to the SQUIRREL model, where the agent observes the state of the environment S_j, mainly how satisfied the current group is. Then it selects an appropriate action α_j to aggregate the lists B_u^j. This results in the transitioning of the model to the next state S_{j+1} where we update the overall satisfaction of the users and the calculated reward R_{j+1}. Finally, the model returns to the group the generated group recommendation list GL_G^j. For ease of readability, Table 1 describes all the notations used in this work.

3. The SQUIRREL Model

In this section, we go into details for every major component of the SQUIRREL model, namely state, action and reward. These components are flexible and can be adjusted to the needs of the user. For this work, to demonstrate our model we utilized elements from previous works in group recommender systems [8, 10, 11]. Namely, to define the state of the model, we use *user utility scores* that describe the satisfaction of each user individually with the group recommendations. For the reward, we focus more on the group as an entity with *group utilities* scores which showcase how relevant are the recommended data items for the entire group. Finally, we utilize a number of different aggregation functions as the actions of the model.

3.1. State Definition via Users Utility Scores

One of the most essential parts of the SQUIRREL model, is the definition of the state of the environment. The state is the component that describes the current status of the group members. Accordingly, we need a state that is focused on each group member individually. This will help the model to make the best possible choice in the policy, without accidentally disregard any group member, something that is probable if we only consider the group in its entirety. It is often the case where a more generalized picture, will disguise individual needs.

In order to describe how satisfied a user is with the group recommendation list, we calculate two different utility scores. Their *satisfaction* with the proposed items and their *disagreement* with the rest of the group. As in a typical group recommender, we apply a recommendation algorithm for each group member u and generate their corresponding individual recommendation list B_u^j at round j. We consider the user's individual recommendation list, as their ground truth, making SQUIRREL more flexible, by allowing it to consider the single user recommendation system as black-box. Note that since after a recommendation round we augment the system with additional ratings, the ground truth of each user changes between rounds.

Let $B_{u,k}^j$ be a list that contains the k items with the highest scores in the B_u^j, and GL_G^j be the group recommendation list that also contains k items. To define a user's satisfaction with the items in the group recommendation list GL_G^j, we will directly compare them to the best items for the user u, i.e., the $B_{u,k}^j$ list. This is commonly referred as the individual loss with respect to the group recommendations [12]. Specifically:

$$sat(u, GL_G^j) = \frac{\sum_{d \in GL_G^j} p_j(u,d)}{\sum_{d \in B_{u,k}^j} p_j(u,d)}. \tag{3}$$

where $p_j(u,d)$ defines the score of the data item d as it appears in the list B_u^j. That is, the recommendation score of item d for user u at round j, as it has been calculated by a single user recommendation algorithm. In the nominator, we calculate the user's satisfaction based on the group recommendation list. For every item in GL_G^j, we sum the score as they appear in each user's top-k list. The denominator, calculates the ideal case for the user, by simply sum the scores of the k top items in the user's individual recommendation list. This way, we are able to normalize the user's satisfaction score. Note that we do not use the scores as they appear in the group list,

but as they appear in the individual preference list of the user. Since the aggregation phase of the group recommendation process somewhat distorts the group members' individual opinions, we opt to take into consideration only the personal preference scores of each group member.

To demonstrate how satisfied a user is after a sequence of μ iterations, we define the overall satisfaction of a user. Meaning, that this score demonstrates how satisfied the user is with the performance of the system after multiple rounds of recommendations.

Definition 1 (Overall Satisfaction)**.** The overall satisfaction of user u with respect to a sequence $\mathcal{GR}$ of μ rounds of recommendations is the average of the satisfaction scores after each round:

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

After each round, we update the internal state of the model to be the new overall user satisfaction. For each group member, we keep their overall satisfaction up to the current round.

3.2. Reward Definition via Group Utility Scores

The reward that is generated by an action given the state of the environment, is the only mechanism that the model has to determine if an action it took was an appropriate one. This reward can be very flexible in its definition based on what we want our model to achieve. Commonly, we want our model to propose items that are relevant to the entire group. For this, we need to define the *group utility scores*, that portrait how good are the recommendations for the group as a whole.

One option to define the reward is via the group satisfaction score. We generalize the user satisfaction score to describe the satisfaction of the entire group. This score will indicate how well the system is able to balance the individual needs of the group members. A high group satisfaction score means that the system is able to identify items that are relevant to the majority of the group members, and alternatively, a low group satisfaction score indicates that the system has failed in this task.

Specifically, we define the group G's satisfaction concerning a group recommendation list GL_G^j to be the average of the satisfaction of the users in the group:

$$groupSat(GL_G^j) = \frac{\sum_{u \in G} sat(u, GL_G^j)}{|G|}. \tag{5}$$

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

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

This overall group satisfaction can be used for expressing the reward achieved at recommendation round j by action a.

$$R_s(\mathcal{GR}^j) = groupSatO(\mathcal{GR}^j) \tag{7}$$

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

However there is a drawback to the definition of the group satisfaction score, that is we consider the average of the group members' individual satisfaction scores. This can lead us to somehow ignore the dissatisfaction of a user. If all members of the group are highly satisfied with just one exception, a user that has low satisfaction scores, the group satisfaction score will still be high and is probable that the least satisfied user will be ignored. This observation leads us to define a new user utility score, namely the user disagreement. We define the disagreement of a user u as the difference between the u's satisfaction score and the maximum satisfaction score among the group members.

$$userDis(u, GL_G^j) = max_{\forall_{u_l \in G}} sat(u_l, GL_G^j) - sat(u, GL_G^j). \tag{8}$$

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

We can generalize this in a group score by considering at each recommendation round the lowest and highest satisfaction scores among the group [12].

$$groupDis(GL_G^j) = max_{u \in G} sat(u, GL_G^j) - min_{u \in G} sat(u, GL_G^j). \tag{9}$$

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

$$groupDisO(\mathcal{GR}) = max_{u \in G} satO(u, \mathcal{GR}) - min_{u \in G} satO(u, \mathcal{GR}). \tag{10}$$

As a second alternative reward of the SQUIRREL model, we can utilize the harmonic mean of $groupSatO$ and $groupDisO$, namely their F-score. Considering the input functions that F-score needs, we use $1 - groupDisO$ to simulate the group agreement.

$$R_{sd}(\mathcal{GR}^j) = 2\frac{groupSatO(\mathcal{GR}^j) * (1 - groupDisO(\mathcal{GR}^j))}{groupSatO(\mathcal{GR}^j) + (1 - groupDisO(\mathcal{GR}^j))}. \quad (11)$$

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

In this work, we utilize these two different reward functions, namely the reward function R_s which is based on the overall satisfaction of the group and the reward function R_{sd} which is based on the overall satisfaction and disagreement. However, the reward function is very flexible and can be altered to complement the specific purpose that the framework is used for.

3.3. Actions as Aggregation Methods

The actions are the driving force behind our model. The agent observes the state of the environment and the history of the rewards it has already achieve and decides to apply an action. The action will generate changes to the state which will then enable us to calculate a reward. Since the actions are the only mechanism that can effect change, they are the natural selection to implement the group recommendation process. As we have already mentioned, we employ a single recommender system to generate recommendation list for each group member. For our model, we consider this process as a black box. The group recommendation process can then be distilled down to aggregating these distinct recommendation lists into one group recommendation list. We consider different approaches for this with each one being a different action.

We consider the following 6 different aggregation methods for the SQUIRREL model. Specifically:

Average. It is the classic Average Aggregation method, where the group predicted score for an item is the average across all predicted scores for that item across all group members. In this case, all the predicted scores of an item for the group members are considered to be of equal importance.

RP80 [10]. This method combines group relevance and group disagreement scores. The authors define group relevance, $avg(d, G)$, of an item

d to be the average prediction score that was produced from the single recommendation system across all the members of group G. They define the group disagreement, $dis(d, G)$ for item d to be Average Pair-wise Disagreement between the predicted scores of the group members. $dis(d, G) = \frac{2}{|G|(|G|-1)} \sum(|p(u_i, d) - p(u_l, d)|)$, where $u_i, u_l \in G$, and $u_i \neq u_l$. The Average Pair-wise Disagreement reflects the degree of consensus in the relevance scores for the data item d among group members. The final score for d for the group G is: $\mathcal{RP}80(d, G) = (1 - w) * avg(d, G) + w * (1 - dis(d, G))$, where w is a tuning parameter for the group relevance and disagreement. Overall, in this case, all the group members' predicted scores about an item are considered of equal importance and additionally, the method takes into account the group's disagreement about that item.

Par [11]. This method generates a group recommendation list by incrementally adding to it items that have the best combination of Social Welfare (SW) and Fairness (F) scores. The Social Welfare of an item is the average satisfaction of all members for that item. [11] uses a similar satisfaction measure as our own work (Equation 5), if the group recommendation list GL_G^j only consisted of one item, d. Thus, $SW(d, G) = groupSat(d)$. As Fairness, $F(d, G)$, it employs the variance over the member's satisfaction scores. The final score of an item for the group is: $\mathcal{PAR}(d, G) = \lambda * SW(d, G) + (1 - \lambda) * F(d, G)$. Overall, this method takes into account the average satisfaction that an item produces for the group members and balances it with the variance of the group members' satisfaction for that item.

SDAA [8]. The Sequential Dynamic Adaptation Aggregation method, at round j, dynamically computes a weight w, which is defined as the difference in the satisfaction scores in the previous round of recommendations between the most satisfied and the least satisfied group member, $w_{SDAA}^j = max_{u \in G} sat(u, GL_G^{j-1}) - min_{u \in G} sat(u, GL_G^{j-1})$. This weight is utilized to compute the final group prediction score for each proposed item d, as a weighted sum between the item's average score across all group members at round j, $avgG(d, G, j)$ and the item d predicted score for the least satisfied user in the group: $score(G, d, j) = (1 - w_{SDAA}^j) * avgG(G, d, j) + w_{SDAA}^j * leastG(d, G, j)$. The function $avgG(G, d, j)$ returns the average score across all group members for item d at round j. The function $leastG(G, d, j)$ returns the score for d for the least satisfied user at the beginning of round j. That is, the least satisfied user after round $j - 1$. When $j = 1$, i.e., the first recommendation round, the aggregation function turns into a simple average aggregation

function. Overall, this method takes into account the past satisfaction of the group members to calculate a weight. This weight balances the average predicted score of an item for the group and the least satisfied member's predicted score for that item.

SIAA [8]. Similarly to SDAA, the Sequential Individual Adaptation Aggregation method also utilizes a weight for aggregations. In contrast to SDAA, which considers the group as an entity, SIAA focuses on each group member individually. For each member, it calculates a weight based on their overall satisfaction and the user disagreement of the previous iteration: $w^j_{SIAA}(u) = (1-b)*(1-satO(u, \mathcal{GR}^{j-1}))+b*userDis(u, GL^{j-1}_G)$, where b is the weight we use to balance the overall satisfaction and user disagreement scores. Given that we are in the j-th round of recommendations, $\mathcal{GR}^{j-1}$ expresses the recommendations of all the previous $j-1$ iterations. We apply this weight directly to the predicted score for each item after the single recommendation process is done. The final group prediction score for an item is the average over all group members' weighted scores. Overall, this method calculates a weight for each group member and applies it during the aggregation phase. The weight is based on the overall satisfaction of the user and their satisfaction on the previous round of recommendations.

Avg+ [8]. From evaluations performed in previous works, we observed that the classic average aggregation function offers one of the best group satisfaction scores, but simultaneously one of the worst group disagreement scores. In order to counter this drawback, we propose the Avg+ aggregation method that consists of two phases. In the first phase, we employ an average aggregation. Then, in the second phase, we iteratively populate the group recommendation list, with items that generate the minimum possible group disagreement score: $GL^j_G = GL^j_G \cup \{d \mid min_{\forall d \in AvgList^j_G}(groupDis(GL^j_G \cup d)) \vee d \notin GL^j_G\}$, where $AvgList^j_G$ is the list of the top k items after the first phase of the aggregation.

In this work, we study the above mentioned aggregation methods as our actions in the model, however they can be augmented by any additional method.

3.4. Problem Formulation

Given the state, actions and reward we have defined, the purpose of our model is the following. At the j^{th} round of recommendations, the model is given as input a group G and a set of recommendation lists B^j_u, for all $u \in G$

generated by a single user recommendation system for the current round. The model consists of the following: (1) a state S - each group member overall satisfaction up to the $j-1$ round, (2) a set of actions A - the different aggregation methods that the model has available, and (3) a reward function R - either based on the overall group satisfaction or the combination of group satisfaction and group disagreement.

At the j^{th} round and all former and subsequent rounds of recommendations, the model considers the state of the group, and selects an action a that maximizes the expected chosen reward. Then, the selected action is applied to aggregate the set of recommendation lists B_u^j to a group recommendation list GL_G^j and further transition the state from s^j to s^{j+1}.

4. Experimental Set Up

4.1. Datasets

For the evaluation section of our work, we did not have access to datasets that contain interactions between groups and a system, where for example, the group as an entity has rated an item[1]. In lieu of such a dataset, we artificially created groups based on information taken from three real datasets. The 20M MovieLens Dataset [14], the 15M book reviews from GoodReads [15], and the Amazon dataset [16]. MovieLens contains 20M ratings across 27,3K movies given by 138,5K users between 01.1995 and 03.2015. The GoodReads dataset contains around 15M ratings from 465K users for about 2M books. Finally, the Amazon dataset contains 18M reviews given by 1.1M users spanning from May 1996 to July 2014.

Since we want to evaluate our model for a sequence of recommendations rounds, we need to simulate a time flow, where between the rounds some time has passed. This means that the recommender does not start with all the data available, but the information is sequentially augmented after each round. To simulate this, we first sort each dataset chronologically by the time that each rating was given. Then we split the datasets into chunks and after each round a new chunk is introduced to the system.

MovieLens Dataset. Initially, we divide the dataset almost evenly into two parts. The first part is the starting dataset of the system, and

[1] An experimental method that can be used for observing the group decision-making process appears in [13].

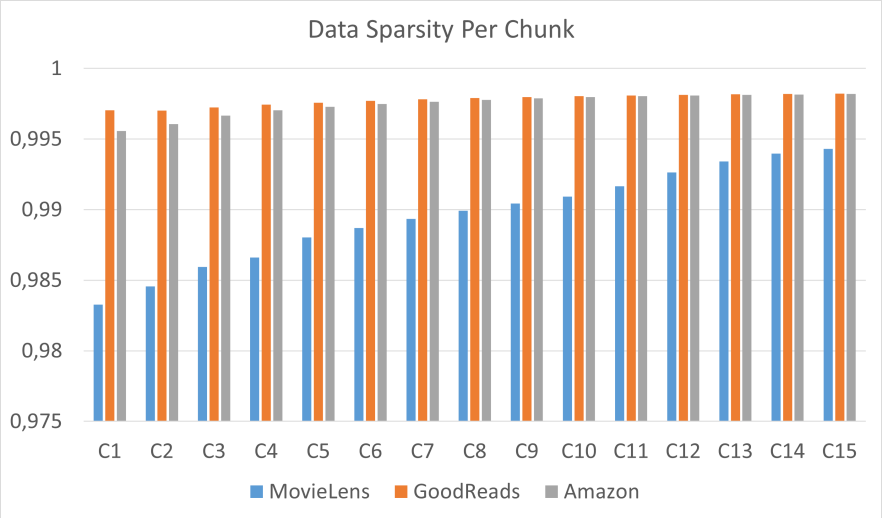


Figure 2: The sparsity of the three datasets, MovieLens, GoodReads and Amazon, when each chunk is augmented into the system.

contains the chronologically first ratings, which were given between 01.1994 and 12.2003. This initial dataset consists of 8.381.255 ratings, 73.519 users and 6.382 movies. The reason we initiate the system with such a large dataset is to avoid the cold start problem[2]. The second part of the dataset is further split into chunks based on timestamps. We create 22 chunks, where each one includes information for a period of 6 months. Of those 22 chunks, we utilized the first 14 for our experiments.

GoodReads Dataset. Due to the difference in the distribution of ratings in the GoodReads dataset, we used a slightly different splitting process, since we have very few ratings in the first years and numerous in the latter. The initial dataset for the MovieLens was almost the 40% of the whole dataset. We initiate the system with a proportional sized data for the GoodReads, which translates into around 6.296.000 ratings. We split the rest of the dataset into chunks of equal size, which contain 674.565 additional ratings each.

Amazon Dataset. An additional pre-processing step is needed for the

[2]The cold start problem in recommender systems appears when there is not enough information about a user and the system is unable to propose to him/her relevant items. This problem is beyond the scope of this research.

Amazon dataset. Unlike the MovieLens and GoodReads datasets, in the Amazon dataset the minimum number of ratings that a user has reported is five (5), whereas in the former are twenty (20). We further prune the dataset and exclude all the users that have rated less than 20 items. This results in a dataset that consists of 10.5M ratings from 180.118 users for 722.003 items. We split the Amazon dataset following the same process as the GoodReads dataset. The initial Amazon set consists of roughly the 40% of the overall dataset, which is translated to roughly 4M ratings. The rest of the dataset is split equally to 14 chunks. Finally, unlike MovieLens and GoodReads, Amazon contains duplicate ratings, meaning that the same user has rated the same item more than once. We only keep the most recent rating in this case.

All three datasets are split evenly in terms of their overall size. The initial chunk is equivalent to their 40% of the entire size and the rest 60% is split into 14 chunks. However, all three datasets differ in terms of their sparsity (i.e., how many ratings are missing over $|I \times U|$). Figure 2 shows the sparsity of all three datasets with C1 being the initial chunk. The sparsity of each subsequent chunk is computed after it has been augmented into the system, with C15 being the entire dataset. The sparsity of the datasets increases after the inclusion of each chunk, because we insert into the system additional new users and items with relatively low number of ratings that correspond to either (i.e., the number of ratings given by a new user and the number of ratings a new item receives). It is worth noting that the GoodReads and Amazon datasets are far more sparse than the MovieLens, leading to have an adverse effect on the quality of recommendations (we discuss this in Subsections 5.4).

4.2. Group Formation

Our goal is to evaluate our model using some real life scenarios. Since we do not have information, for example, which users are friends or have liked a review given by a user, we presume that if two people share the same interests, then they have similar ratings for the same data items. We want to simulate two different real-life scenarios. First, when a new person enters an established group, and their interests may very well be dissimilar to the rest of the group. For example, consider a new friend that joins an already established group in a movie nights series. Second, consider the case when random people join together for an activity, like a book club, where each member has their own tastes, however, they will have to read the same book.

We evaluate our model on these two types of groups based on the different degree of similarity shared between the group members. We utilize only the initial datasets from the splitting process to calculate the similarity scores and form groups. We use the Pearson Correlation [17] similarity function, which takes values from -1 to 1. Higher values imply a higher similarity between the users, while negative values indicate dissimilarity:

$$s(u_i, u_l) = \frac{\sum_{d \in X}(r(u_i, d) - \bar{r}_{u_i})(r(u_l, d) - \bar{r}_{u_l})}{\sqrt{\sum_{d \in X}(r(u_i, d) - \bar{r}_{u_i})^2}\sqrt{\sum_{d \in X}(r(u_l, d) - \bar{r}_{u_l})^2}}$$, where X is a set that consists of the items that are rated by both users, $r(u, d)$ is the rating that user u gave to item d and $\bar{r}_u$ is the mean of u's ratings.

We regard two users to be similar if they have an 0.5 similarity score or above, and dissimilar when they have -0.5 or lower similarity score. The types of groups we are considering are the following:

4 similar – 1 dissimilar (4+1): The four members of the group are similar to each other, while the single member is dissimilar to all the rest.

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

4.3. Single User Recommendations

In our evaluation, we derive the group recommendation list by combining the individual recommendation lists of each group member. We consider this first step, the single recommendation, as a black box, and our system recognize the individual recommendation lists of each user as that user's ground truth, since this is the only information that it has for that user. The actual single user recommendation system can be any system available, and is not restricted in any fashion beyond the fact that it needs to generate a recommendation list for a user. Systems that generate only one item can be used, but then the aggregation phase is rather trivial.

For this work, we use a user-based Collaborative Filtering (CF) recommender [18]. It generates a recommendation list by first finding similar users and then exploiting their likes and dislikes to propose items. The similarity function we used in CF is once again the Pearson Correlation. We consider two users similar if they share a similarity score greater than a threshold and they have given a rating to at least 5 common items. Due to the difference in each dataset's semantics, we set different similarity thresholds

for each dataset. The MovieLens similarity threshold is 0.8, and for the GoodReads and Amazon dataset is 0.7[3]. Given a user u, to predict preference scores for item d, we use the Weighted Sum of Others Ratings [19], $p(u,d) = \bar{r}_u + \frac{\sum_{u' \in (P_u \cap U(d))} s(u,u')(r(u',d) - \bar{r}_{u'})}{\sum_{u' \in (P_u \cap U(d))} |s(u,u')|}$. Since a user may have many similar other users, we only consider the top 100 most similar to them denoted as P_u and $U(d)$ describes all the users that have rated item d.

5. Experimental Results

5.1. Preliminaries

In our experiments, at each round, we propose to the group a list of 10 recommended items. Given that the group is aware of all data items presented to the group in a previous round, we do not recommend these items again[4].

We also use the two format of groups that correspond to two life scenarios. (1) 4 similar – 1 dissimilar **(4+1)**, and (2) 5 dissimilar **(5Diss)**. For each type, we generate 100 different groups with five members from each dataset, MovieLens, GoodReads and Amazon. From those, we use 80 groups as a training set for our model and the remaining 20 groups as the test dataset. Additionally, we use the test dataset to individually evaluate all the aggregation functions we use as actions, namely Average, SDAA, SIAA, Avg+, RP80 and Par.

Training the model for only a specific group type is rather limiting since it will force the model to learn only that model. However, there is a case to be made about the need of such an approach. For example, if there is an application that groups people randomly for an activity for more varied social interactions, then a system that specialises in balancing the different needs of people that are dissimilar to each other is more advantageous. Nonetheless, we want our model to be able to be utilized in more universal conditions. To achieve this, we randomly selected 40 groups from the training sets for each

[3]Due to the fact that MovieLens is more dense than GoodReads and Amazon, we selected a higher similarity threshold for it, enabling Pearson Correlation to return approximately the same number of similar users for all datasets.

[4]Alternatively, we can imagine that the group is presented with one item only. Given that we target at the group consensus, the prediction model regarding the group choice for this item is a worth studying problem without a straightforward solution that we leave it for future work.

group type and 10 from their corresponding test sets. This generated three additional training and test sets, one for each dataset, MovieLens,GoodReads and Amazon, with 80 groups in the training set and 20 groups in the test set, denoted as the **AllGroups** test set. To avoid over-tuning the model for a specific group type, during the training phase we randomized the order of the groups.

To create our SQUIRREL RL model, we use the Tensorforce library[5], with the Proximal Policy Optimazation (PPO) as our learning policy, with probability to transition from state s to state s' under a random action is set to 0.3, while the γ parameter is set to 0.99. Since each dataset is fundamentally different from the other, notably the Goodreads and Amazon datasets are far more sparse than the MovieLens (Figure 2), we experimentally found the best learning rate, namely, the speed at which the model *learns*, for each dataset individually. This was done through extensive experiments that we do not present due to space constrains. We used 0.0022 learning rate for the MovieLens dataset and 0.0020 for the GoodReads and Amazon.

Furthermore, each individual aggregation method has further parameters to tune. For the RP80 aggregation function, we choose to set the variable w to 0.8, since it performs better based on the experiments presented in [10] for the MovieLens dataset, which we also use. This weight is used in balancing the group relevance and group disagreement. According to [11], the Par method uses $\lambda = 0.8$ to combine the social welfare and fairness scores. In [8], for the SIAA aggregation function, we empirically found, using the same datasets, the best value for variable b, which is used to combine the overall satisfaction and disagreement of a user, is 0.2. Finally, in the same work, for Avg+, we found that the best results are given when $k = 50$, where k is the number of items with the highest prediction score.

5.2. Measures

At each round, we propose to the group a list of 10 recommended items. At the end of each recommendation round, we calculate the average of $groupSatO(G, \mathcal{GR})$ and $groupDisO(G, \mathcal{GR})$ for all the groups in the test set. These measures are used to calculate the reward functions R_s and R_{sd}.

To further analyze the methods' quality, we use as secondary measure the Normalized Discounted Cumulative Gain (NDCG), where we want the best

[5]https://tensorforce.readthedocs.io/en/latest/

possible items for a user to appear with the highest ranking possible in the group recommendations:
$NDCG(u, G, j) = \frac{DCG(u,G,j)}{IDCG(u,G)}$, with $DCG(u, G, j) = \sum_{d \in GL_G^j} \frac{|d \cap B_u^j|}{log_2(r(d,GL_G^j)+1)}$, where $r(d, GL_G^j)$ is the rank of item d in the group recommendation list GL_G^j. IDCG is the best possible outcome for user u_i. We supplement the results from NDCG, with the Discounted First Hit (DFH) metric that counts if a user has seen an item that is highly relevant to him/her in the early ranks of the group recommendation list: $DFH(u, GL_G^j) = \frac{1}{log_2(fh+1)}$, where fh is the rank of the first item that appears both in the group recommendation list and the individual preference list of the user. Since NDCG and DFH refer to one user, we apply them to the group by computing them for each member and then taking the average over them, as recent papers on group recommendations do (e.g., [20]). We do this at the end of each round.

5.3. Analyzing the Actions Selected

In Tables 2 and 3, we present the actions that the model selected during both the training and testing phase for both versions of the reward function. Table 2 presents the actions selected when utilizing the reward function R_{sd}, which is based on satisfaction and disagreement (Equation 11), while Table 3 shows the actions selected when we use the satisfaction reward function R_s (Equation 7).

With either reward function SQUIRREL selects the most appropriate aggregation to apply, based on what maximizes that reward function. In more detail, in Table 2 showcasing the actions tested and selected for the reward function R_{sd} for the MovieLens dataset, we can observe that the model has predominantly chosen the SDAA aggregation function with minimal exceptions. We can explain this choice with Figures 4, 5 and 6, where we show the R_{sd} values for all aggregation methods for the MovieLens dataset. SDAA is the best action for the model to take, since it produces the best R_{sd} scores due to its lower group disagreement scores in the latter rounds.

It is also worth to examine the number of times the model did not select SDAA. For all test sets, the model does not select the SDAA function in favor of the Avg+ function, exactly 20 times (see Table 2). These numbers correspond with the number of groups present in each test set. This is significant because the R_{sd} scores for the first round are comparable between many aggregation functions but the Avg+ function perform slightly better since it has lower group disagreement scores. Again we can corroborate this,

MovieLens Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	50	0	70	0	69	0
SDAA	869	280	917	280	387	280
SIAA	69	0	47	0	317	0
Avg+	122	20	105	20	315	20
Par	71	0	54	0	93	0
RP80	19	0	7	0	19	0
GoodReads Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	28	0	7	0	165	0
SDAA	22	0	35	0	24	0
SIAA	65	20	227	20	216	20
Avg+	860	280	676	280	590	280
Par	190	0	51	0	144	0
RP80	35	0	204	0	61	0
Amazon Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	149	0	193	0	127	0
SDAA	64	0	222	0	106	0
SIAA	333	240	356	240	504	240
Avg+	242	60	296	60	216	60
Par	199	0	56	0	87	0
RP80	213	0	77	0	160	0

Table 2: Actions chosen during the training and testing phase for all group formats, while utilizing the reward function based on satisfaction and disagreement, R_{sd}.

MovieLens Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	83	20	48	0	130	0
SDAA	22	0	90	20	132	0
SIAA	366	0	26	0	140	280
Avg+	7	0	25	0	553	0
Par	701	280	940	280	187	20
RP80	21	0	71	0	58	0
GoodReads Dataset						
	4+1		5Diss		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	32	0	20	0	319	269
SDAA	229	20	278	20	159	11
SIAA	135	0	62	0	143	0
Avg+	35	0	40	0	233	20
Par	659	280	772	280	158	0
RP80	110	0	28	0	188	0
Amazon Dataset						
	4+1		5Dis		AllGroups	
	Train	Test	Train	Test	Train	Test
Avg	412	220	237	20	533	270
SDAA	47	0	166	0	96	0
SIAA	163	0	517	280	184	0
Avg+	59	0	27	0	129	0
Par	382	80	192	0	235	30
RP80	137	0	61	0	23	0

Table 3: Actions chosen during the training and testing phase for all group formats, while utilizing the satisfaction reward, R_s.

when we examine the R_{sd} scores generated by all aggregation methods shown in Figures 4, 5 and 6. This is because two of the proposed aggregation functions, SDAA and SIAA, do not have the necessary information during the first round of recommendations. SDAA defaults to a classic Average for the first round and SIAA utilizes the user disagreement of the previous round, something that does not exist during the first round. So both of them default to a classic average aggregation for the first round. On the other hand, Avg+ does not have any such drawback. The model is able to identify this, and selects the better performing aggregation function for the first round.

SQUIRREL behaves similarly when utilizing the GoodReads dataset as well. As we can observe in Figures 8, 9 and 10, the Avg+ aggregation function offers the best R_{sd} scores out of all the aggregation methods used as actions. However, during the first round, many aggregation functions share the same R_{sd} scores, so the system has selected a different action for that round. Amazon presents similar results, where now the aggregation that produces the best R_{sd} scores is SIAA with the exception of the first three rounds where Avg+ is better, as it is shown in Figures 12, 13 and 14. This is why we can see in Table 2 that the Avg+ aggregation method is selected 60 times during the test phase.

Finally, Table 2 also shows that the actions that were selected during the testing phase for the reward function R_{sd} remain the same across all different training sets (i.e., 4+1 and 5Dis). Subsequently, the same action is selected when we train the model using the **AllGroups** training set that contains all different formats of groups.

Table 3 shows the actions taken during the training phase and selected by the SQUIRREL model during the test phase when utilizing the reward function R_s. As we can see in Figure 3[a,b,c]-left for MovieLens, Figure 7[a,b,c]-left for GoodReads, and Figure 11[a,b,c]-left for Amazon, the satisfaction generated by the aggregation functions are approximately the same with two notable exceptions. First, RP80 is designed to offer the best possible group disagreement in lieu of the group satisfaction, thus it has lower satisfaction scores. Second, Avg+ is based on the Average method, which offers one of the best group satisfaction scores. However, Avg+ also considers sub-optimal items (in regards to the group satisfaction score they produce) so as to discover items with also good group disagreement.

The similar satisfaction scores of the various aggregation methods make it difficult for our model to select an aggregation function that is universally good for each dataset. This is because R_s considers just the overall group

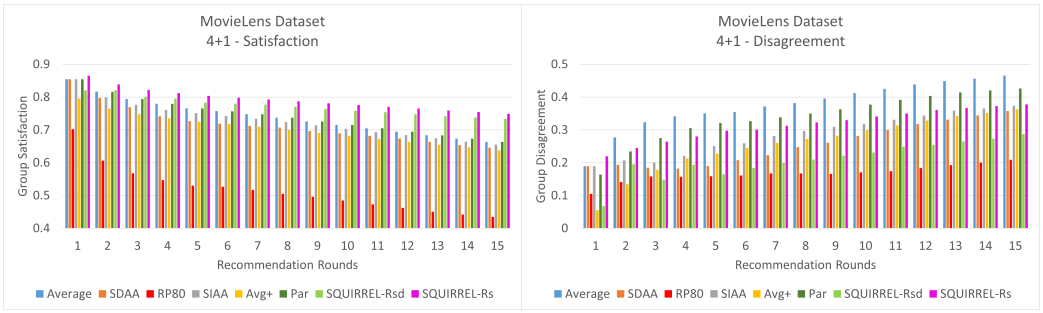
satisfaction, and multiple aggregation functions (i.e., actions) produce very high satisfaction scores. This results in different functions to be selected for the various training sets. In contrast, R_{sd} is more refined, as it considers both group satisfaction and group disagreement, and is able to identify the most effective aggregation function for each dataset (see Table 2).

5.4. Analyzing the Satisfaction and Disagreement Scores

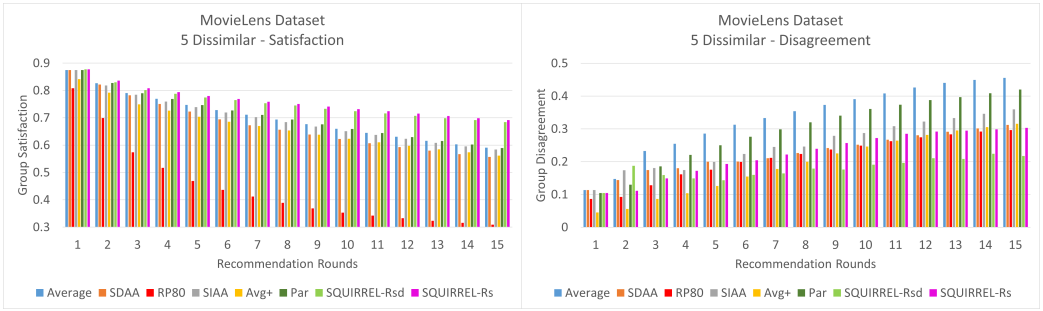
The SQUIRREL model primarily selects one aggregation function, the one that maximizes the selected reward function, but is also more flexible during the first round of recommendations. Due to this small change that the model makes, the primary aggregation function (the one that the model uses for the rest of the rounds) performs differently. For example, when we compare the results in Figure 3a that shows the group satisfaction and group disagreement for the 4+1 test set for the MovieLens dataset, and Figure 4 that shows the R_{sd} scores for all aggregation methods, we can see that SQUIRREL correctly identifies that the primary aggregation function (in this case the SDAA) under-performs in the first round, so it selects the best one which is the Avg+. In this case, SDAA underperforms in the first round of recommendations, since we do not have the previous disagreement of the users and so the aggregations defaults to the classic Average. This selection of a different aggregation method in the first round has a cascade effect on the results for the rest of the rounds. We can observe a similar behaviour for the rest of the test sets.

In general, the small change on the first round, makes the model perform better for the rest of the rounds both for group satisfaction and group disagreement. In terms of group satisfaction SQUIRREL offers the best group satisfaction scores. This difference is mainly due to first round of recommendations, where as already stated, the model chooses a different aggregation function than the one typically used in the rest of the recommendation rounds. Thus, it now has a different pool of items to recommend to the group in the next rounds. Subsequently, this makes it behave differently than the aggregation method that is primarily used in the rest of recommendation sequence. This is more apparent in the latter rounds of recommendations, where SQUIRREL is able to perform better than the rest of the aggregation methods.

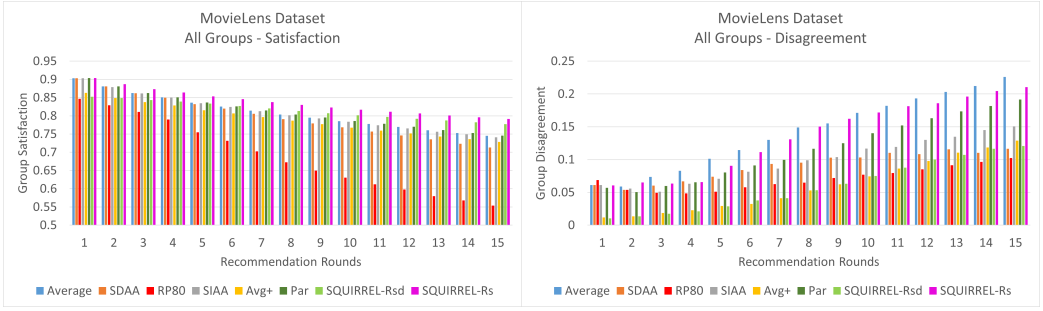
When comparing the results between the two reward functions, for example the last two bars in Figure 3[a] for the MovieLens dataset and the 4+1 training set, the R_s reward function offers better group satisfaction



(a)



(b)



(c)

Figure 3: MovieLens Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

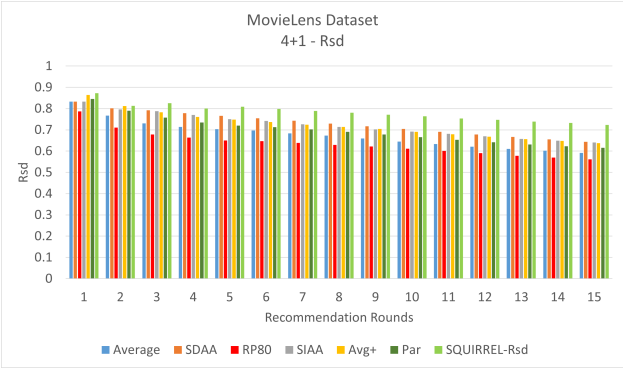


Figure 4: MovieLens Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

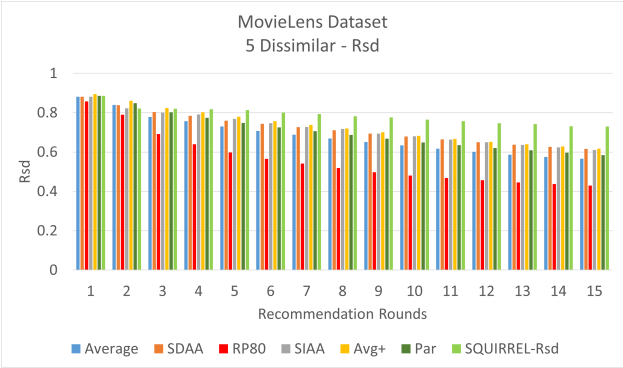


Figure 5: MovieLens Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

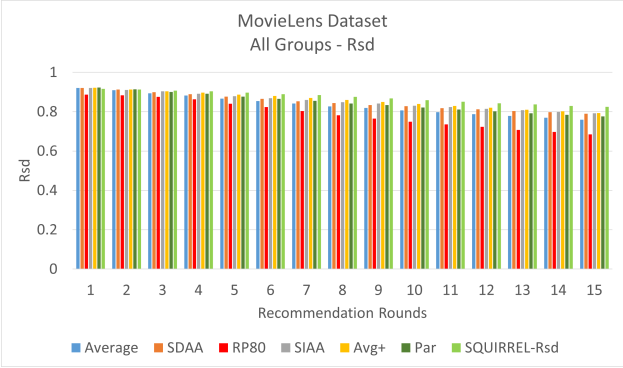
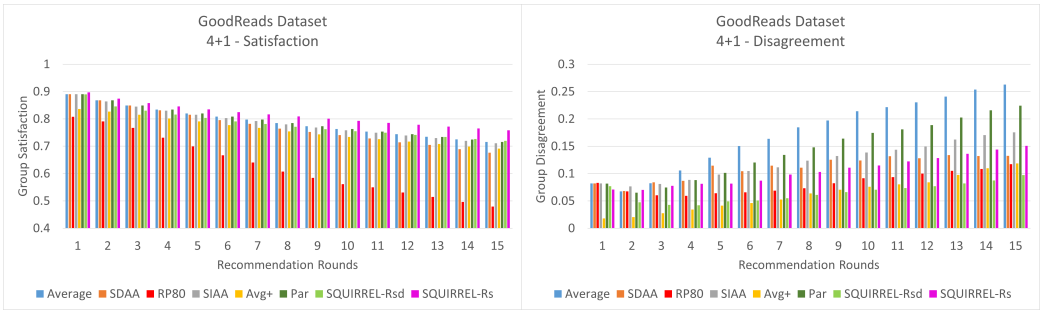
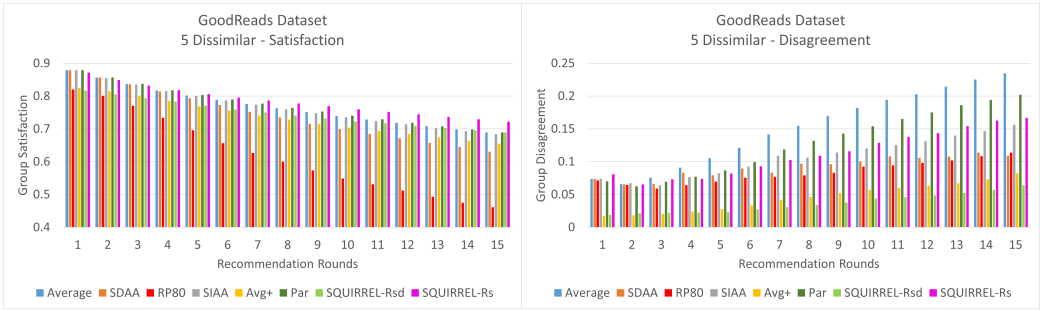


Figure 6: MovieLens Dataset: R_{sd} scores for all aggregation methods under the AllGroups test scenario.



(a)



(b)

(c)

Figure 7: GoodReads Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

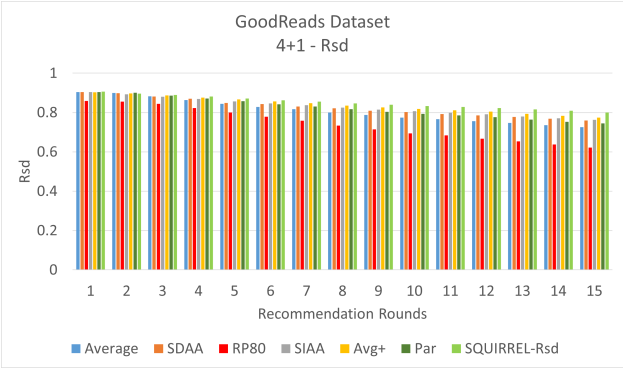


Figure 8: GoodReads Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

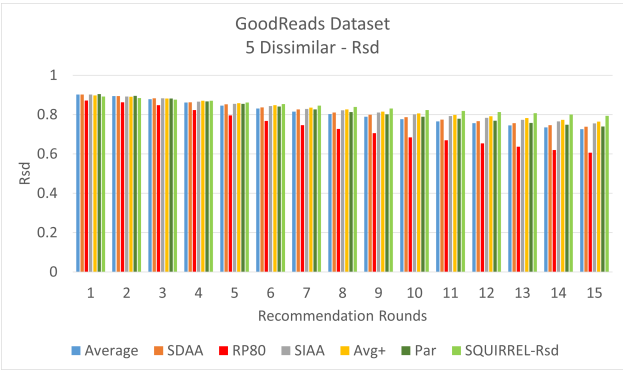


Figure 9: GoodReads Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

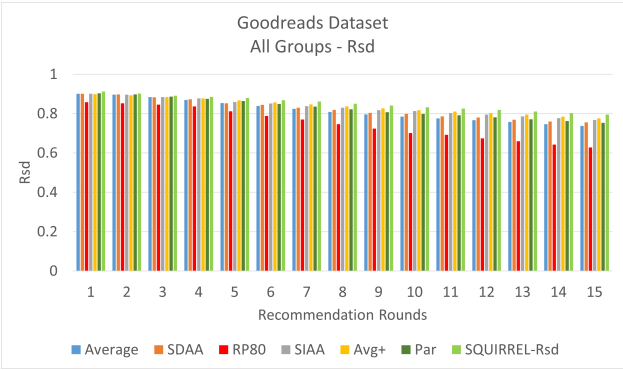
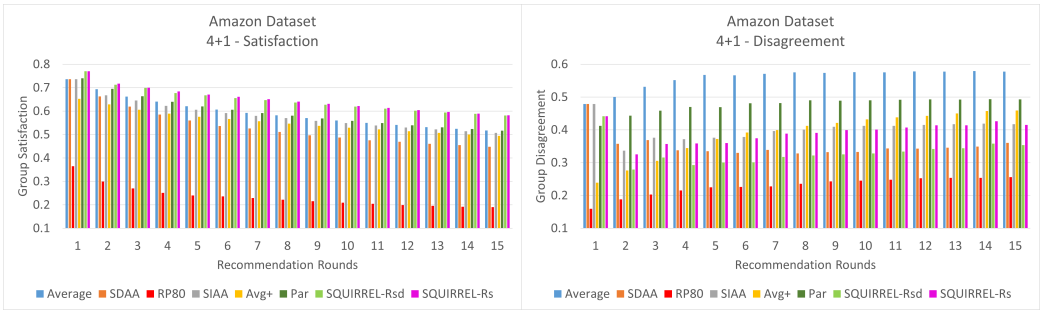
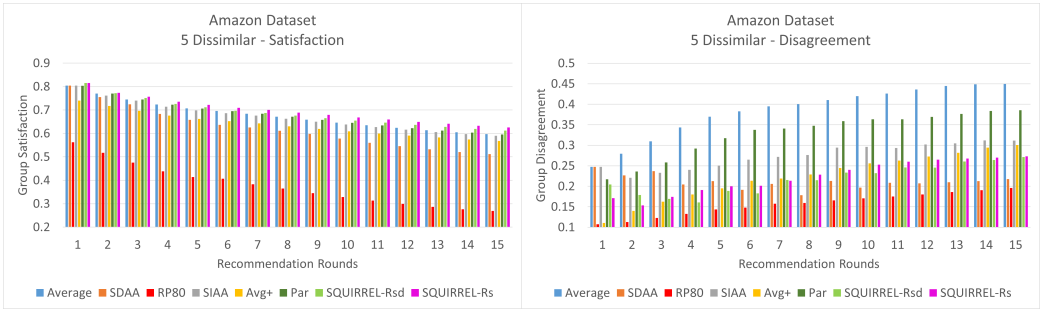


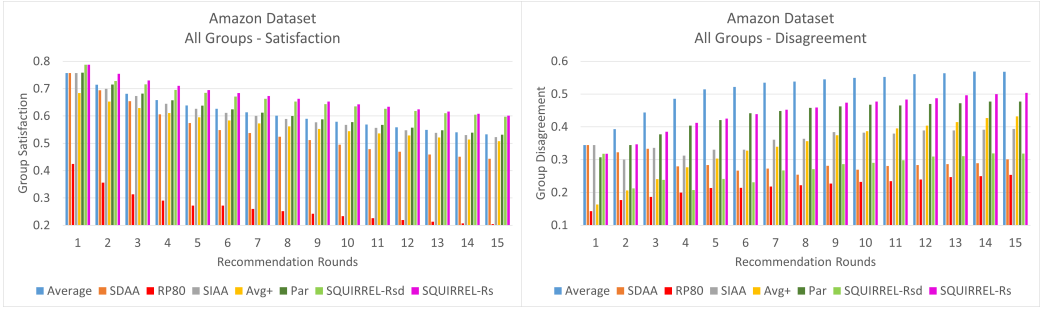
Figure 10: GoodReads Dataset: R_{sd} scores for all aggregation methods under the All-Groups test scenario.



(a)



(b)



(c)

Figure 11: Amazon Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods and all test scenarios, (a) 4+1, (b) 5Diss and (c) AllGroups.

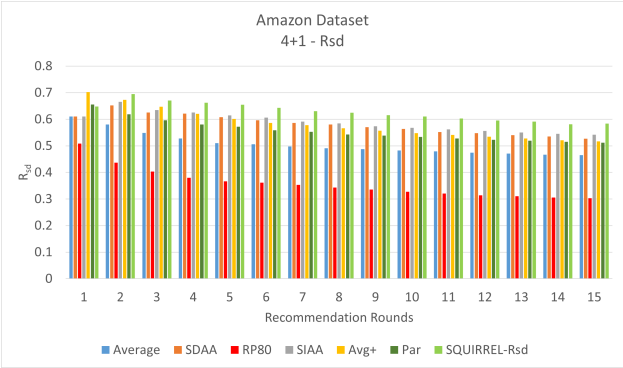


Figure 12: Amazon Dataset: R_{sd} scores for all aggregation methods under the 4+1 test scenario.

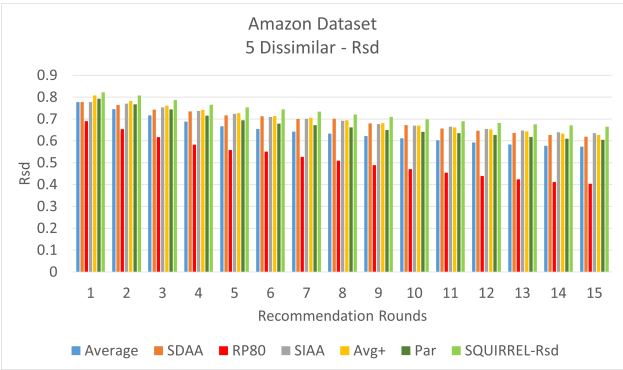


Figure 13: Amazon Dataset: R_{sd} scores for all aggregation methods under the 5Diss test scenario.

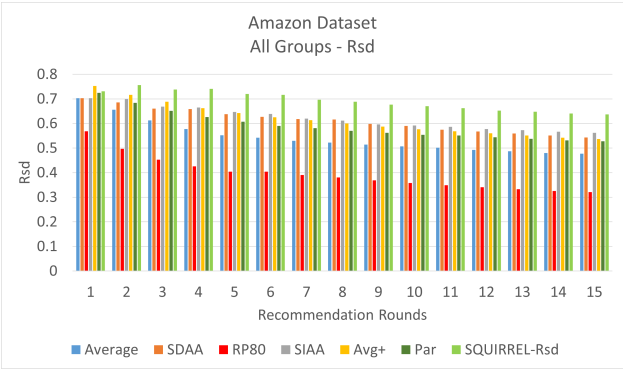


Figure 14: Amazon Dataset: R_{sd} scores for all aggregation methods under the AllGroups test scenario.

scores (Figure 3[a]-left), since that reward function maximizes group satisfaction. However, it generates worse group disagreement compared to R_{sd} (Figure 3[a]-right). This is expected, since the R_{sd} considers both group satisfaction and disagreement. When we compare the rest of the aggregation functions, for group satisfaction SIAA, SDAA, Par and Average have very comparable results, while the Avg+ falls slightly behind since it also considers sub-optimal items in terms of the group satisfaction they generate. On the other hand, RP80 offers very low group satisfaction scores, but one of the best group disagreement scores. This is because, it was designed to minimize the disagreement score. Additionally, RP80 has better group disagreement in more homogeneous group formats like 4+1 while it falls slightly behind in 5Diss test set. SDAA and SIAA have very good group satisfaction but average group disagreement scores. On the other hand, Average and Par have very good group satisfaction scores but also very high group disagreement.

We can observe that the aggregation methods have different behaviours in the three different datasets. For example, the best performing one for MovieLens is SDAA, for GoodReads is Avg+ and for Amazon dataset is SIAA. The discrepancy of performance between the three datasets is because the difference in their sparsity, namely GoodReads and Amazon datasets are more sparse than the MovieLens. This negatively affects the single recommender system and reduces the number of items that are highly relevant to each individual group member. On the one hand, in the MovieLens dataset which is the most dense dataset of the three (see Figure 2), the single recommender system generates more relevant items per group member. This benefits the SDAA aggregation, since it directly balances the group satisfaction and group disagreement. Thus the more common items that are relevant to most group members the more beneficial it is for the aggregation method.

On the other hand, the sparsity of the GoodReads and Amazon datasets benefits the Avg+ and SIAA aggregation methods. A lower number of highly relevant items per user means that during Avg+'s first phase where it selects the top 50 items with the highest Average score, all the best possible items for the group are considered. Then, Avg+ can select the items that produce the best group disagreement. SIAA only takes into consideration each individual group member's satisfaction and disagreement scores and not the group's, thus is better equipped to handle the sparsity of the datasets. The SQUIRREL model is able to adjust to the different datasets and outperform all the individual aggregation methods.

Finally, we can see from Figures 3, 7 and 11 that for all test scenarios the

MovieLens Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.038	0.041	0.038	0.033	0.039	0.021	0.054	0.052
5Diss	0.043	0.044	0.043	0.036	0.043	0.017	0.054	0.060
AllGroups	0.042	0.044	0.041	0.034	0.042	0.019	0.058	0.057
GoodReads Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.041	0.053	0.043	0.032	0.042	0.020	0.046	0.063
5Diss	0.042	0.057	0.042	0.031	0.043	0.017	0.040	0.065
AllGroups	0.042	0.053	0.042	0.030	0.042	0.020	0.043	0.066
Amazon Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.028	0.047	0.035	0.025	0.033	0.003	0.044	0.026
5Diss	0.030	0.052	0.031	0.024	0.030	0.005	0.029	0.024
AllGroups	0.032	0.056	0.036	0.030	0.035	0.003	0.052	0.043

Table 4: NDCG values for all datasets, MovieLens, GoodReads and Amazon and all test scenarios, 4+1, 5Diss and AllGroups.

performance of all aggregation methods is reduced after each round of recommendations. This is because of two main reasons. First, as Figure 2 shows, the sparsity of all datasets is increased after each additional chunk of data is augmented in the system. Second, after each round, we eliminate the top 10 most suggestions for the group, i.e., these items cannot be recommended again.

5.5. Quality of Recommendations

In Table 4, we show the NDCG and in Table 5 the DFH scores for all aggregation functions, as well as, for the SQUIRREL model when utilizing the two reward functions. We present the average scores after the end of all 15 rounds of recommendations. The NDCG scores in Table 4 reflect how good the recommendations were for the group, i.e., how many items in the group recommendation list are also included in the user's individual recommendations. The DFH scores in Table 5 describe how relevant at least one item in the group recommendation list is for the users.

In Table 4, we see that the SQUIRREL model's higher NDCG values mean that it is able to identify more relevant items for the groups. Also it recommends items that are highly ranked by each user, given that the DFH scores are high. In contrast, the Avg+ aggregation method has lower NDCG scores, because it also considers items that may not be highly relevant to each user. This is further corroborated by the group satisfaction scores shown in

MovieLens Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.900	0.868	0.897	0.936	0.913	0.810	0.923	0.937
5Diss	0.886	0.819	0.871	0.932	0.899	0.743	0.914	0.940
AllGroups	0.915	0.875	0.905	0.945	0.925	0.807	0.930	0.936
GoodReads Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.968	0.923	0.967	0.982	0.977	0.936	0.957	0.987
5Diss	0.962	0.893	0.958	0.979	0.968	0.929	0.954	0.973
AllGroups	0.976	0.929	0.975	0.985	0.982	0.955	0.964	0.985
Amazon Dataset								
	Average	SDAA	SIAA	Avg+	Par	RP80	SQUIRREL-R_{sd}	SQUIRREL-R_s
4+1	0.795	0.679	0.779	0.838	0.825	0.554	0.832	0.935
5Diss	0.886	0.755	0.876	0.916	0.903	0.714	0.890	0.935
AllGroups	0.820	0.677	0.804	0.860	0.844	0.579	0.837	0.872

Table 5: DFH values for all datasets, MovieLens, GoodReads and Amazon and all test scenarios, 4+1, 5Diss and AllGroups.

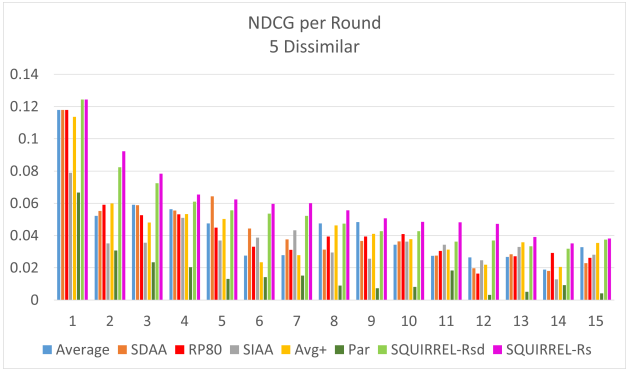


Figure 15: MovieLens Dataset: NDCG values per recommendation round for the 5Diss test scenario.

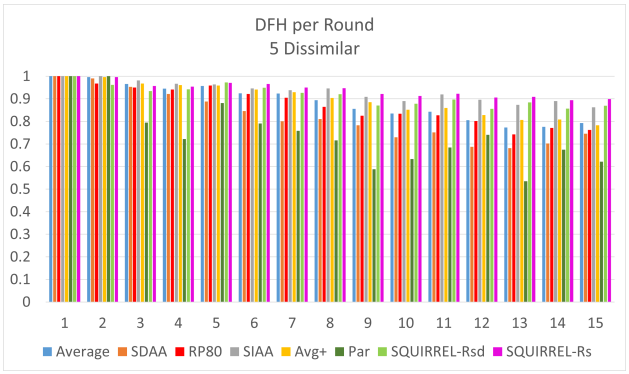


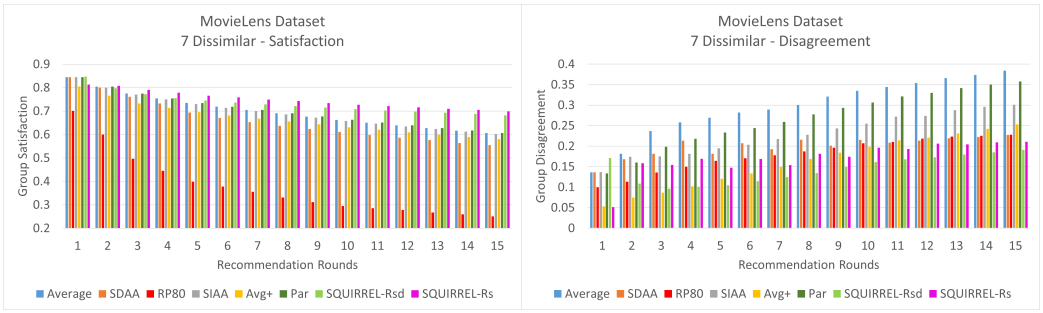
Figure 16: MovieLens Dataset: DFH values per recommendation round for the 5Diss test scenario.

Figures 3-left, 7-left and 11-left. The aggregation methods that have high satisfaction scores, like Average, SDAA, SIAA, Par and both SQUIRREL models also have high NDCG scores. This is because both the group satisfaction score and NDCG describe the same general principle of how relevant are the items in the group recommendation list for each user.

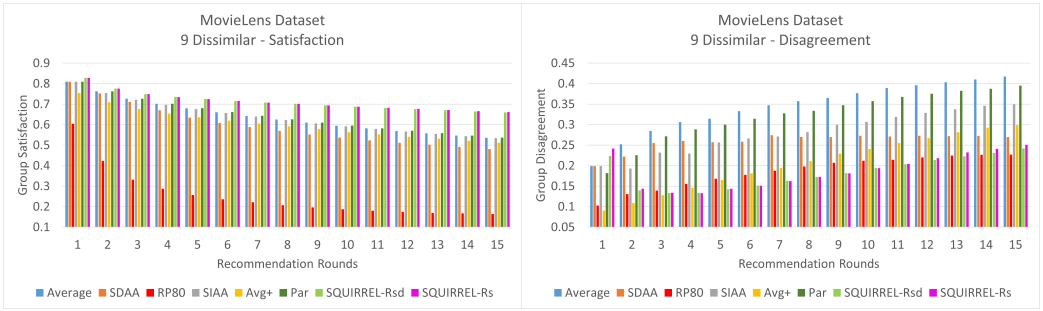
Finally, to better visualize the results, Figures 15 and 16 show the NDCG and DFH values, respectively, per round. We showcase the results for the MovieLens dataset and the 5Diss test scenario. These values are the average for all groups in the 5Diss test set per round. The reward function R_s has the best overall NDCG values since it also has the best group satisfaction (Figure 3b-left). Additionally, it is more apparent that during the first round of recommendations all aggregation methods have a higher performance which is then significantly reduced in the following rounds. As already stated, this is because, as shown in Figure 2, the sparsity of all datasets is increased after each chunk is augmented in the system, as well as, because after each round we eliminate from consideration the top 10 items for each group.

5.6. Group Size Evaluation

For all the previous evaluations, the group size was set to five. Next, we evaluate our model when the group size is increased. We selected the MovieLens dataset to experiment with since its higher density allowed us to form larger groups. We expand on the 5 dissimilar test scenario and introduce two new test cases, 7 dissimilar (**7Diss**) and 9 dissimilar (**9Diss**). Figure 17a shows the group satisfaction (left) and group disagreement (right) for



(a)



(b)

Figure 17: MovieLens Dataset: Group Satisfaction (left) and Group Disagreement (right) for all aggregation methods under the 7Diss and 9Diss test scenarios.

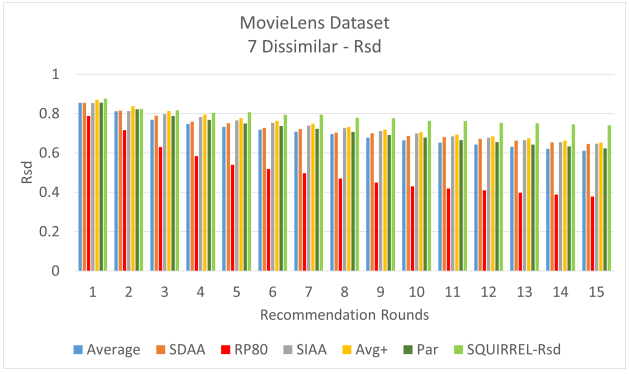


Figure 18: MovieLens Dataset: R_{sd} scores for all aggregation methods under the 7Diss test scenario.

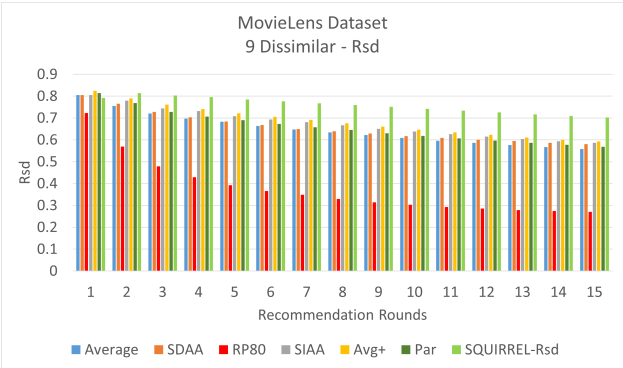


Figure 19: MovieLens Dataset: R_{sd} scores for all aggregation methods under the 9Diss test scenario.

the 7Diss test scenario. Figure 17b shows the respective scores for the test scenario 9Diss.

The performance of all aggregation methods lowers when the group size is increased. This is expected since it is more difficult to satisfy a larger group. However, SQUIRREL, overall, is able to more effectively handle a large number of dissimilar group members. It is largely unaffected by the increased group size with the fall in performance between the 7Diss and 9Diss test cases being minimal. These results are further corroborated when we examine the Figures 18 and 19 showcasing the R_{sd} scores for the various aggregation methods. All aggregation methods have similar results in the first round of recommendation with SQUIRREL outperforming the rest in the latter rounds.

6. Related Work

6.1. Group Recommendations.

Group recommendations have a significant research background [21]. In general, there are two main approaches for group recommendations [22]. In the first approach (e.g., [23, 24]), we combine the group members ratings to form a virtual user so that a standard recommendation approach can be applied. The second approach, and the most popular for group recommendations (e.g., [10, 25, 26]), is to employ a standard single-user recommender system and apply it to each individual group member. Then, we aggregate the group members lists into one single group recommendation list. In this

work, we follow the second approach, since it is more flexible [27] and offers opportunities for improvements in terms of effectiveness.

In the aggregation stage, a group recommender system can take into consideration many different criteria. For example, work done in [28] suggests a group recommendation model which takes into account each individual group member's influence during the aggregation phase. They state that the more knowledgeable a member is about the items considered for recommendation, the more influence they have, meaning they have a higher weight during the aggregation phase. Additionally, [29] draws from recent advancements in attention network and neural collaborative filtering to deduce the aggregation strategy from the available data. In the same vein, [30] in addition to an attention mechanism, it also employs a Bipartite Graph Embedding Model (BGEM) to infer each member's influence to the group's final choice. [31] uses a preference-oriented social network and the social interactions among the group members, to finalize the group's choice without having access to complete preferences of the members.

By observing how group members interact with one another, [32] determines the best aggregation strategy. These interactions were modeled as multiple voting processes in order to simulate how a consensus is reached, and a stacked social self-attention network was proposed to learn the voting scheme of the group members. In dividing a large group of people into subgroups based on their own interests, [33] offers a novel method of producing recommendations for a large group. Specifically, it identifies a set of potential candidate media-user pairs for each subgroup and aggregate the CF recommendations lists for each such pair. [34] proposes a two-phase group recommender that, tries to satisfy all the group members. In the first phase, they try to satisfy the whole group. In the second phase, they try to satisfy the members individually by filtering out irrelevant items to each member.

In [11], each group member is assigned a utility score based on how relevant the recommended items are to them. Then it balances the utility of the group members and generates a group recommendation list. In [20], the utility of a user is defined by the similarity between the individual and group recommendations of the user. Their approach involves considering sets of N-level Pareto optimal items when creating the group recommendation list. As part of the aggregation phase, [35] proposes a notion of rank-sensitive balance. As far as possible, the first recommendation should balance the interests of all group members. Similarly, the first two items together must also do the same; and so on.

Any of these rank aggregation methods can be modified to work on the SQUIRREL model as an additional action. Depending on the requirements of the method, supplementary changes should be made to the model, such as a different state or reward definition or additional input should be given to the model. Additionally, depending on the complexity of the aggregation method, the overhead of retraining them can be large; a complex group recommendation method needs more time for training than the simple Average aggregation method. In future work, we aim to further develop our model so as other than rank aggregation methods for group recommendation can be readily included as actions.

6.2. Sequential Recommendations.

It is generally considered that sequential recommenders fall into three broad categories, according to how many past interactions they take into account: *Last-N interactions-based recommendations, Session-based recommendations* and *Session-aware recommendations* [36]. The first approach considers only the most recent N user actions [37, 38, 39]. This is due to the amount of historical data logged by the system for the user - many of which are duplicates as well as not providing any useful information - and as a result the system is overwhelmed. The only interactions that are taken into account when making session-based recommendations are the ones that the user performed during the current session. The most common places to find them are in the news and advertisements [40, 41]. In the last category, the system has information about the last interaction between it and the user, as well as the user's history. E-commerce and app recommendations often employ these recommenders [42, 43, 44]. Another session-aware music recommendation system is proposed in [2]. Based on a neural network architecture, users' preferences are represented as a sequence of embeddings, one for each session. The user's recent selections and session-level context (such as device usage and time) can predict the tracks a user will listen to. [3] proposes a multi-round recommender system using Variational Autoencoders (VAEs) and introduces randomness into the regular operation of VAEs to achieve fairness [45] during multiple rounds, while in order to minimize bias and promote diversity, [4, 46, 47] penalizes scores given to items based on historical popularity.

The framework in [1] uses cross-neighbor relation modeling to uncover collaborative information using a bipartite graph, where the users and items are depicted as nodes and the interactions between them as the links. They

not only consider the directly linked nodes but also the 2-hop neighbours as well, which they call high-order collaborative relations. They utilize them and both user-side and item-side historical sequences to capture user and item dynamics more effectively. The work in [48] proposes the GLS-GRL system, where an item-item co-occurrence graph captures user-item interactions in the entire history, as well as an item-item co-occurrence graph containing the same information for the current time period. As a result of graph representation learning, the GLS-GRL system is able to achieve long-term and short-term representations of users and subsequently merge them to obtain integrated representations of users. A constraint-based user-interactive attention mechanism encodes relationships between group members into group representations used for recommendations.

The work presented here, employs methods for sequential group recommendations, that were first proposed in [49, 8]. SDAA considers the group as a whole, and dynamically calculates a weight based on the satisfaction of the group members. This weight is then used to combine two scores; the average preference score of an item for all group members and the preference score of the item for the user that is the least satisfied in the previous round of recommendations. In contrast, SIAA considers each group member individually. At each round it calculates a weight for each user based on the user's overall satisfaction and the user's disagreement in the previous round. Finally, Avg+ capitalizes on the advantages of the classic Average method while simultaneously tries to minimize its drawbacks. Overall, the previous works mentioned concern recommendation systems for single users.

Under specific application examples, [50] and [51] perform empirical research to investigate different aggregation strategies for suggesting a sequence of television items and music tracks, respectively, to groups of users. The proposed framework in this work is the first to focus on selecting an aggregation strategy, among a pool of available ones, for each round of group recommendations, relying on reinforcement learning.

6.3. Reinforcement Learning in Recommendations

In recent years, more and more research is focused in utilizing reinforcement learning for recommendations [52]. One of the first works in this domain was done in [53], where they propose a web recommender system, where the state of the environment is the last N pages that the user visited, actions are the page recommendations and reward is a weighted sum between the ranking of the recommended page and the time the user spend on that page.

[54] proposes a DQN-based reinforcement learning framework for online personalized news recommendation. The framework is composed of two parts, offline and online. During the offline stage, the model is trained and during the online stage the agent produces a recommendation and logs the user feedback. After a period of time, the model enters again in the offline stage and depending on the logged feedback may make changes to the model. The system can accurately model the dynamic features of the news, in addition to the user preferences, in order to increase reward in the long term. In addition to click/no-click feedback, they also consider user return pattern in order to determine the user behavior, and employ an exploration strategy for more recommendation diversity. [55] employs reinforcement learning techniques to optimize the recommendation model for long-term accuracy of recommendations. They consider two main areas; cold-start and warm start. The model relied on the interactions between the recommender system (environment) and users (agents). This allowed it to be applied to environments with inadequate content information.

The work done in [56] proposes a list-wise recommendation framework based on deep reinforcement learning, a method that reduces redundant computation in scenarios with a large and dynamic item space. They train and evaluate the model offline, while applying the recommender system online. To successfully evaluate the model in offline stage, they simulate the response of a user (reward) using a user-agent interaction environment simulator, where the reward is calculated based on the similarity between the current state and the action taken with other historical data. [57] proposes a deep learning movie recommender model based on reinforcement learning, which utilize prioritized experience replay to depict the changes in the interests of the user over time. The system models the recommendation process as reinforcement learning, and in order to recommend movies based on user preferences, they use agents to learn about users' interests and movie features.

The work done in [58] proposes a user-side system for sequential music recommendation. It combines in a Markov Decision Problem (MDP) the users' explicit and implicit feedback. The explicit feedback consists of the users' music channel preferences and the implicit is generated by the users when they request a new music track. An MDP is also used in [59] where they propose a commercial system that utilizes an ordered sequence of selections for each user as the state of the environment, to predict and recommend a new item. However, since the e-commerce environment of this model is different than a classic recommendation system, several assumptions are made in order

to be readily deployed as an application.

Each of these works offers a solution to the recommendation problem through reinforcement learning, but in most cases are centered around a specific domain of recommendations. In our work, we propose a framework that aspires to be more flexible in terms of the domain it can be applied and is able to combine different techniques to counter-balance the drawbacks that are inherently present in each recommendation solution.

7. Conclusion

In this paper, we propose the SQUIRREL framework, which enables sequential group recommendations via reinforcement learning. We treat the single user recommendation system as a black-box, and focus on aggregating the individual group members' recommendation lists into a group list. Specifically, we focus on producing group recommendations via rank aggregation, thus the actions that the model can take are various rank aggregation methods. The state of the model is the overall satisfaction of the individual group members, and we examine two different reward functions. The model is able to be further configured; additional rank aggregation methods can be applied, a different state can be defined and further reward functions can be considered. The model is able to dynamically select the most appropriate group recommendation method to apply depending on the state of the group.

We used three real-world datasets, 20M MovieLens, GoodReads and Amazon, to evaluate not only our proposed model but additionally, all the individual aggregation methods used as actions. The different semantics of each dataset alter the performance of all methods, but typically do not alter their performance comparative to each other. We train our model using different formation of groups, which depict different scenarios for group recommendations. Additionally, we combined the different group types into one test set to simulate a more general scenario. Although the actions selected for each group type are different, the overall performance of the SQUIRREL model did not change between the test cases. The SQUIRREL model is able to correctly identify the aggregation method that best maximizes the reward function utilized. We corroborate this with extensive evaluation of all methods used as actions in our model, and test it with two reward functions. Additionally, the model recognizes the drawbacks of certain aggregation methods and counters them by selecting different ones at the opportune moments. This has as a result to completely alter and enhance the performance of the

primary aggregation method used. Finally, we evaluated the quality of the recommendations provided by calculating the NDCG and DFH values for the SQUIRREL model and all the aggregation methods used as actions. We show that SQUIRREL is able to provide quality recommendations, with one of the best NDCG and DFH scores.

In our future work, we want to extend our model by combing individual user and group satisfaction scores to define new states and rewards. Moreover, we want to expand the SQUIRREL model so as to be able to consider as actions other group recommendation algorithms in addition to the rank aggregation methods. This will require a restructure on the input that the model receives and further optimizations, depending on the complexities of the selected group recommendation methods.

References

[1] J. Qin, K. Ren, Y. Fang, W. Zhang, Y. Yu, Sequential recommendation with dual side neighbor-based collaborative relation modeling, in: WSDM, 2020.

[2] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, M. Lalmas, Contextual and sequential user embeddings for large-scale music recommendation, in: RecSys, 2020.

[3] R. Borges, K. Stefanidis, Enhancing long term fairness in recommendations with variational autoencoders, in: MEDES, 2019.

[4] R. Borges, K. Stefanidis, On mitigating popularity bias in recommendations via variational autoencoders, in: SAC, 2021.

[5] J. Masthoff, Group Recommender Systems: Combining Individual Models, Springer US, Boston, MA, 2011, pp. 677–702.

[6] K. J. Arrow, A difficulty in the concept of social welfare, Journal of Political Economy (1950).

[7] R. D. Burke, M. Ramezani, Matching recommendation technologies and domains, in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer, 2011, pp. 367–386.

[8] M. Stratigi, E. Pitoura, J. Nummenmaa, K. Stefanidis, Sequential group recommendations based on satisfaction and disagreement scores, Journal of Intelligent Information Systems (2021).

[9] E. Triantaphyllou, Multi-Criteria Decision Making Methods: A Comparative Study, Vol. 44, 2000. doi:10.1007/978-1-4757-3157-6.

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

[11] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, M. Shaoping, Fairness-aware group recommendation with pareto-efficiency, in: RecSys, 2017.

[12] T. N. Nguyen, F. Ricci, A. Delic, D. Bridge, Conflict resolution in group decision making: insights from a simulation study, User Modeling and User-Adapted Interaction 29 (5) (2019) 895–941.

[13] A. Delic, J. Neidhardt, T. N. Nguyen, F. Ricci, An observational user study for group recommender systems in the tourism domain, Information Technology & Tourism (2018).

[14] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (4) (2015) 19:1–19:19.

[15] M. Wan, R. Misra, N. Nakashole, J. McAuley, Fine-grained spoiler detection from large-scale review corpora (2019). arXiv:1905.13416.

[16] R. He, J. McAuley, Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering, in: WWW, 2016.

[17] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: An open architecture for collaborative filtering of netnews, in: CSCW, 1994.

[18] C. Desrosiers, G. Karypis, A Comprehensive Survey of Neighborhood-based Recommendation Methods, Springer US, Boston, MA, 2011, pp. 107–144.

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

[20] D. Sacharidis, Top-n group recommendations with fairness, in: SAC, 2019.

[21] J. Masthoff, Group recommender systems: Aggregation, satisfaction and group attributes, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 743–776.

[22] A. Jameson, B. Smyth, Recommendation to Groups, Springer-Verlag, 2007, p. 596–627.

[23] J. F. McCarthy, T. D. Anagnost, Musicfx: an arbiter of group preferences for computer supported collaborative workouts, in: Proceedings of the 1998 ACM conference on Computer supported cooperative work, 1998, pp. 363–372.

[24] Z. Yu, X. Zhou, Y. Hao, J. Gu, Tv program recommendation for multiple viewers based on user profile merging, User modeling and user-adapted interaction 16 (1) (2006) 63–82.

[25] L. Baltrunas, T. Makcinskas, F. Ricci, Group recommendations with rank aggregation and collaborative filtering, in: RecSys, 2010.

[26] E. Ntoutsi, K. Stefanidis, K. Nørvåg, H.-P. Kriegel, Fast group recommendations by applying user clustering, in: International conference on conceptual modeling, Springer, 2012, pp. 126–140.

[27] M. O'connor, D. Cosley, J. A. Konstan, J. Riedl, Polylens: A recommender system for groups of users, in: ECSCW, 2001.

[28] Q. Yuan, G. Cong, C.-Y. Lin, Com: A generative model for group recommendation, in: KDD, 2014.

[29] D. Cao, X. He, L. Miao, Y. An, C. Yang, R. Hong, Attentive group recommendation, in: SIGIR, 2018.

[30] H. Yin, Q. Wang, K. Zheng, Z. Li, J. Yang, X. Zhou, Social influence-based group representation learning for group recommendation, in: ICDE, 2019.

[31] A. Salehi-Abari, C. Boutilier, Preference-oriented social networks: Group recommendation and inference, in: RecSys, 2015.

[32] L. Vinh Tran, T.-A. Nguyen Pham, Y. Tay, Y. Liu, G. Cong, X. Li, Interact and decide: Medley of sub-attention networks for effective group recommendation, in: SIGIR, 2019.

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

[34] J. K. Kim, H. K. Kim, H. Y. Oh, Y. U. Ryu, A group recommendation system for online communities, International Journal of Information Management (2010).

[35] M. Kaya, D. Bridge, N. Tintarev, Ensuring fairness in group recommendations by rank-sensitive balancing of relevance, in: RecSys, 2020.

[36] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Comput. Surv. 51 (4) (2018) 66:1–66:36.

[37] C. Cheng, H. Yang, M. R. Lyu, I. King, Where you like to go next: Successive point-of-interest recommendation, in: International Joint Conference on Artificial Intelligence, 2013.

[38] D. Lian, V. W. Zheng, X. Xie, Collaborative filtering meets next check-in location prediction, in: WWW, 2013.

[39] Q. Liu, S. Wu, L. Wang, T. Tan, Predicting the next location: A recurrent model with spatial and temporal contexts, in: AAAI Conference on Artificial Intelligence, 2016.

[40] F. Garcin, C. Dimitrakakis, B. Faltings, Personalized news recommendation with context trees, CoRR abs/1303.0665 (2013).

[41] B. Hidasi, M. Quadrana, A. Karatzoglou, D. Tikk, Parallel recurrent neural network architectures for feature-rich session-based recommendations, in: RecSys, 2016.

[42] N. Hariri, B. Mobasher, R. Burke, Context-aware music recommendation based on latenttopic sequential patterns, in: RecSys, 2012.

[43] D. Jannach, L. Lerche, M. Jugovac, Adaptation and evaluation of recommendations for short-term shopping goals, in: RecSys, 2015.

[44] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: RecSys, 2017.

[45] E. Pitoura, K. Stefanidis, G. Koutrika, Fairness in rankings and recommendations: an overview, VLDB J. 31 (3) (2022) 431–458.

[46] R. Borges, K. Stefanidis, F2VAE: a framework for mitigating user unfairness in recommendation systems, in: SAC, 2022.

[47] R. Borges, K. Stefanidis, On measuring popularity bias in collaborative filtering data, in: EDBT/ICDT Workshops, 2020.

[48] W. Wang, W. Zhang, J. Rao, Z. Qiu, B. Zhang, L. Lin, H. Zha, Group-aware long- and short-term graph representation learning for sequential group recommendation, in: SIGIR, 2020.

[49] M. Stratigi, J. Nummenmaa, E. Pitoura, K. Stefanidis, Fair sequential group recommendations, in: SAC, 2020.

[50] J. Masthoff, Group modeling: Selecting a sequence of television items to suit a group of viewers, User Model. User Adapt. Interact. 14 (1) (2004) 37–85.

[51] A. Piliponyte, F. Ricci, J. Koschwitz, Sequential music recommendations for groups by balancing user satisfaction, in: User Modeling, Adaptation, and Personalization, 2013.

[52] M. M. Afsar, T. Crump, B. Far, Reinforcement learning based recommender systems: A survey (2021). arXiv:2101.06286.

[53] N. Taghipour, A. Kardan, S. S. Ghidary, Usage-based web recommendations: A reinforcement learning approach, in: RecSys, 2007.

[54] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, Z. Li, Drn: A deep reinforcement learning framework for news recommendation, in: WWW, 2018.

[55] L. Huang, M. Fu, F. Li, H. Qu, Y. Liu, W. Chen, A deep reinforcement learning based long-term recommender system, Knowledge-Based Systems (2021).

[56] X. Zhao, L. Zhang, L. Xia, Z. Ding, D. Yin, J. Tang, Deep reinforcement learning for list-wise recommendations (2019). arXiv:1801.00209.

[57] Z. Yuyan, S. Xiayao, L. Yong, A novel movie recommendation system based on deep reinforcement learning with prioritized experience replay, in: ICCT, 2019.

[58] O. Moling, L. Baltrunas, F. Ricci, Optimal radio channel recommendations with explicit and implicit feedback, in: RecSys, 2012.

[59] G. Shani, D. Heckerman, R. I. Brafman, C. Boutilier, An mdp-based recommender system., Journal of Machine Learning Research 6 (9) (2005).