TECHNICAL REPORT

TR-CS-NMSU-2021-02-24

Edgar Ceh-Varela
Huiping Cao
Hady W. Lauw **

Department of Computer Science
New Mexico State University
** Sigapore Management University

February 19, 2021

A Detailed Performance Evaluation of Aggregation-based Group Recommender Systems

EDGAR CEH-VARELA, Department of Computer Science, New Mexico State University, USA
HUIPING CAO, Department of Computer Science, New Mexico State University, USA
HADY W. LAUW, School of Information Systems, Singapore Management University, Singapore

Recommender Systems (*RecSys*) provide suggestions in many decision-making processes, such as deciding the items to buy and the news to read. Given that groups of people can perform many real-world activities, e.g., a family decides a movie to watch or a group of friends plans a trip, the need for recommendations for groups has increased. A wide range of Group Recommender Systems (*GRecSys*) has been developed to aggregate individual preferences to group preferences. However, previous *GRecSys* analyze their systems using different types (sizes and cohesion) of groups, and most of such works focus on testing their systems using only one type of item, called experience goods (EG). As a consequence, it is hard to get consistent conclusions about the performance of *GRecSys*.

This study experimentally compares different group recommendation methods by varying possible recommendation settings. *GRecSys* build upon *RecSys* to individual users and aggregate preferences of users in a group. Thus, we first briefly review multiple representative *RecSys*. Then, we present the aggregation strategies and aggregation functions that *GRecSys* commonly use to aggregate group members' preferences. Next, we examine the group formation and the types of items to be recommended by analyzing 179 studies related to *GRecSys*. Finally, we implement and evaluate the performance of *GrRecSys* with various consistent settings. In particular, we test *seven* representative *RecSys* when used for group recommendations with *two* different aggregation strategies, *ten* different aggregation functions, and *two* different types of items on *seven* real-life datasets, which includes a widely used benchmark dataset. We present our comprehensive evaluation results and provide an in-depth analysis of the *GRecSys* performance in different settings.

CCS Concepts: • **Computing methodologies** → *Machine learning*; • **Information systems** → **Collaborative filtering**.

Additional Key Words and Phrases: group recommender systems, aggregation strategies, recommendation scenarios

1 INTRODUCTION

Nowadays, with the explosive growth and variety of information available on the internet, searching for solutions to almost any type of problem is a common practice. However, due to the large amount of information available, and the one that is being generated every day, the task of finding the right solution sometimes could be tedious or even impossible for novel users.

Recommender Systems (*RecSys*) help users get through the problem of information overload [106]. In the Big Data era, the research field of *RecSys* gained increasing interest, attracting efforts from multiple disciplines, including artificial intelligence, data mining, statistics, and human-computer interaction. The main task of a recommender system is to provide suggestions to a user, or group of users, in various decision-making processes, such as deciding the items to buy, the music to listen, or the news to read [21, 92].

Most works on *RecSys* propose methods to make recommendations for individual users. Recently, the number of works dealing with recommendations for groups of people has increased. This increase is because many different real-world activities (e.g., watching a movie, going to a restaurant, planning a trip with friends), are performed by groups. For such group activities, *RecSys* have to suggest relevant recommendations using the individual preferences of group members. Systems that make recommendations for groups of users are called *Group Recommender Systems (GRecSys)*.

Although significant progress has been made in the field of *GRecSys*, many challenges are not yet well addressed. One of these challenges is how *GRecSys* are evaluated, mainly because there is

no clear standard for this task. Each proposed solution establishes its objectives and evaluation metrics, without putting much emphasis on comparisons with baselines considering different usage scenarios. For example, analyze scenarios considering the group's size, group cohesion, and the type of elements to recommend. This situation does not allow a fair comparison between methods, which generates many questions such as which algorithm performs better in different situations, how and when they can be used, and which solution is the best given a particular recommendation scenario (or setting).

We have identified three particular issues regarding the *GRecSys* evaluation. First, there is no defined standard regarding the number of group members. Some works use small groups [23, 72, 79], some use medium-sized groups [2, 32, 54], some use large groups [45, 47, 63, 74], and a few of them use very large groups [2, 61]. The conclusions of these works are sometimes contradictory. For example, some studies conclude that the performance of *GRecSys* in small groups is better [71], while others get the opposite conclusion, i.e., generally, *GRecSys* works worse for small groups [99].

Second, to the best of our knowledge, there is no publicly available dataset with ground truth information regarding the groups and group members and their preferences. Most studies have specific strategies to create user groups. However, there is no standard regarding the characteristics of the group members. Different studies mention they randomly form their groups [18, 24, 69], other studies evaluate groups formed with high cohesion among members (i.e., similar users) [73, 76], while others formed groups with entirely dissimilar members [16, 64], and some used combinations or variations of those previous methods [9, 68, 80]. Not all studies consider all these different types of groups when conducting their evaluations. Moreover, studies using proprietary datasets usually do not indicate how coherent (e.g., similar, dissimilar, random) the members are [51, 78, 96].

Third, in online stores, items can be classified as experience goods (i.e., movies, travels, restaurants) and search goods (i.e., electronics, clothing) [62]. For example, movies are *experience goods* because a user needs to watch the movie to perceive its attributes. In contrast, a digital camera is a *search good* because a user can get its attributes without a previous interaction. These items have features that influence the way users prefer one item over another. The vast majority of research papers use the MovieLens dataset[1] or similar datasets and do not use a dataset of items with different characteristics to contrast their results. The evaluation contrasting different types of items has not been previously investigated.

In summary, the lack of consistent evaluation settings about *GRecSys* performance concerning the size of the groups, the characteristics of group members, and the effectiveness in different types of items, raises certain doubts about in which scenarios a type of *GRecSys* is better than another, or if a particular approach can be used in a different scenario.

More specifically, the following research questions need to be addressed:

(1) How does group fomation (group sizes and groups of different cohesion) influence the performance of *GRecSys*?
(2) Is there a significant difference in performance when *GRecSys* are evaluated using datasets with different types of items?
(3) Depending on the scenario (group size, group cohesion, and item type), what types of *GRecSys* produce better results?

In this work, we explore how *GRecSys* perform when used with user groups of different group configurations and with different product types. The remainder of the article is organized as follows. We first present an overview of *RecSys* and *GRecSys* in Section 2. In Section 3, we present the elements from different group recommendation scenarios, like group sizes, group types, and item

[1] http://grouplens.org/datasets/movielens/

Table 1. Papers inclusion criteria for the systematic review.

	Inclusion Criteria
IC1	The paper proposes a technique to recommend items for a group of users.
IC2	The paper presents a comparison of the proposed technique and other group recommender systems.
IC3	The paper proposes techniques and metrics to evaluate and explain recommendations for a group of users.

types. We describe the experimental settings in Section 4. The results from our tests are presented in Section 5. Finally, in Section 6, we conclude the article.

2 RECOMMENDER SYSTEMS

We conduct an analysis of the literature related to *GRecSys* by going through works published in peer-reviewed venues between the years 2009 and 2020. Three inclusion criteria (IC) were used to select papers for our review. We summarise the inclusion criteria in Table 1. We obtain **179** papers (excluding duplicates) and one book.

2.1 Recommender Systems (RecSys) for individual users

RecSys denote the set of tools and techniques that are used to recommend items to users. Here, an *item* can be anything that is recommended (e.g., movies, books) by *RecSys*. Item recommendations could be personalized. Personalized *RecSys* make a different recommendation to different users, while non-personalized *RecSys* recommend the same items (e.g., the top-10 most popular movies) to all the uses.

According to [14], *RecSys* can be classified into six categories: (i) collaborative filtering (CF), (ii) content-based (CB), (iii) demographic, (iv) knowledge-based, (v) community-based, and (vi) hybrid. Among these categories of *RecSys*, collaborative filtering, content-based, and hybrid types are the most widely utilized. More recently, *Deep Learning-based Recommender Systems (DLRS)* have emerged combining *RecSys* from these three categories and Deep Neural Networks techniques. In what follows, we briefly introduce these methods. The detailed definitions are in Section 2.3.

2.1.1 Collaborative Filtering (CF) recommender systems. This type of *RecSys* is the most widely used approach. To a given user looking for suggestions, CF algorithms use other users' preferences to make recommendations [87]. These systems use past interactions between similar users or similar items to produce new recommendations. These interactions are generally in the form of ratings, which can be explicitly given by the users or implicitly obtained by observing how a user interacts with an item.

CF algorithms are categorized in two ways: *memory-based* and *model-based* [33]. Memory-based approaches use a *user-item* matrix containing ratings that users have given to items in previous interactions. A CF recommender system uses this matrix to predict the ratings of those items the user has not yet interacted with. The most common memory-based strategies are based on *user-user* (user-based) and *item-item* (item-based) similarity [60].

On the other hand, in model-based solutions, a recommender system first analyzes the interactions between users and items. Then, it creates a generative model to explain such interactions. Finally, the obtained model is used to make recommendations.

The main problem of CF approaches is called "cold-start problem" [3]. As CF uses the ratings in the *user-item* matrix to make new recommendations, the cold start problem emerges when the user-item matrix does not contain a sufficient number of entries for a new user or a new item to calculate a recommendation.

2.1.2 Content-based (CB) recommender systems. Contrary to CF, where only the ratings from the user-item matrix are considered, CB approaches use extra information about items' features.

For example, in the movie domain, some features could be the movie genre, its director, or the actors. CB recommenders analyze the features from those items that a user liked in the past to find similar items to recommend [7]. Based on these features, CB systems build item profiles and user preference profiles. With this information, CB systems build a model explaining the interactions between users and items.

CB systems also suffer from the cold start problem, but to a lesser extent than CF systems, mainly because new items can be described based on their characteristics and features. If an item contains features previously unseen, then the cold start problem is still an issue.

2.1.3 Hybrid recommender systems. Hybrid systems combine CF with CB to address the limitations in CF or CB. Many approaches exist for building a hybrid system [13]. The most common strategies are listed below.

- Weighted: the CF and CB predicted ratings are combined into a weighted mean.
- Mixed: CF and CB prediction lists are found separately and then merged into a single list.
- Switched: CF predictions or CB predictions are used depending on specific criteria.
- Feature combination: CF and CB features are considered together to find the most similar users or items.
- Feature augmentation: additional features are used to predict ratings, and the main recommender uses these ratings to produce the recommendation list.

Hybrid systems alleviate the problems of cold-start and data-sparsity of the user-item matrix. Furthermore, the combination of methods can improve the accuracy of recommendations.

2.1.4 Deep Learning-based Recommender Systems (DLRS). In recent years, deep learning (DL) methods have been used to capture non-linear user-item relationships. These methods attempt to obtain low-level representations for the users and items (i.e., embeddings), whereas learning their relationships directly from the data.

These *DLRS* usually integrate DL models with traditional recommender system approaches. For example, in [42], a CB approach is used to capture the relationship from different data sources such as contextual, textual, and visual information. In the same way, CF approaches have been used in [37, 84, 88, 95, 101, 105] to address the sparsity and cold-start problem.

2.2 Recommender systems for groups

The recommender systems recommending items to a group of users is called *Group Recommender Systems (GRecSys)* [59]. Group recommendation is a complex task. First, the preferences of each group member need to be considered. Depending on the group characteristics, conflict may arise when users have different preferences towards the same item or a set of items. Second, many strategies exist to aggregate preferences of individuals. Different strategies may output different recommendations, which could impact the group members' final satisfaction.

2.2.1 Aggregation strategies. Algorithms for *GRecSys* are in many cases based on those algorithms used in individual *RecSys*. Different aggregation strategies, derived from social choice theory, are applied to these algorithms when used for group recommendations. Two basic aggregation strategies exist for *GRecSys* [91]: (i) *individual-recommendations aggregation*, also called aggregated predictions or aggregated ratings (denoted as PRED), (ii) *individual-preferences aggregation*, also called aggregated models or aggregated profiles (denoted as PROF).

Most works [5, 31, 32, 45, 46, 65, 67–69, 81–83, 86, 94, 111, 114] use the PRED aggregation strategy. The basic idea of this strategy is that recommendations are made independently for each group member. Then group recommendations are produced by aggregating the ratings for each of these individual recommendations. Figure 1 shows the stages of this aggregation strategy.

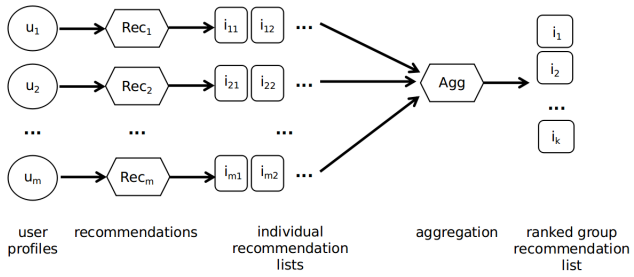


Fig. 1. PRED aggregation strategy

Approaches using the PROF aggregation strategy [27, 56, 72, 80, 89, 108, 112] creates a profile for a *"virtual"* user by aggregating each group member's item preferences, representing the preferences of the group. Then, recommendations are calculated for this *"virtual"* user. Figure 2 shows the stages of this aggregation strategy. Another group of works [11, 22, 40, 66, 102] has combined both

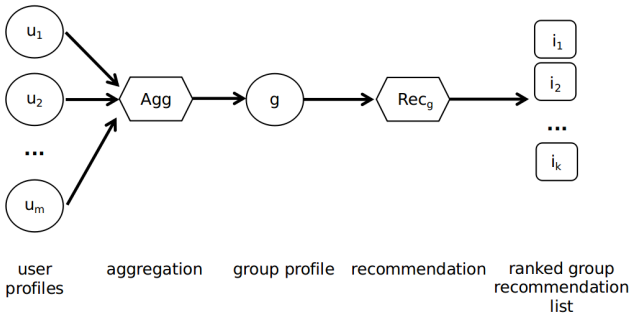


Fig. 2. PROF aggregation strategy

aggregation strategies to get a better group representation.

2.2.2 Aggregation functions. For any of the aggregation strategies mentioned before, a major concern is the generation of group recommendations considering each member's individual preferences. Based on social choice theory, several aggregation functions were presented in [58] and in [26], including *Additive, Approval, Average, Average without Misery, Borda Count, Least Misery, Most Pleasure, Most respected person, Multiplicative,* and *Popularity*. These aggregation functions are formally defined in Section 4.5. Most works use some *"consensus"* functions. In [5, 8, 31, 32, 40, 102, 111] an *Average* function is implemented using the group members' rating lists. A set of works use functions focusing on those *most popular items* (i.e., majority voting). For example, [8, 68, 82, 116] use the *Borda Count* function. Similarly, some aggregation functions take into account only a *subset of user preferences*. The most widely used is the *Least Misery* [2, 8, 31, 32, 40, 73] and the *Most Pleasure* [31, 73] functions.

For different problems and domains, some of these strategies work better than others. Although these strategies are frequently used, there is no formal study to indicate when to use one or the other, or if the combination of any of these can provide better results. In this study, we analyze these strategies for different scenarios.

2.2.3 Models and systems. In many cases, the recommendation method of *GRecSys* is based on models from single-user *RecSys*. The most preferred algorithms in this category are those related to CF approaches. Works like [32, 40, 66, 68, 81, 110, 112] use *Matrix Factorization* algorithms

(e.g., SVD) to extract latent features representing the group members. Other works [5, 56, 67] use popular methods such as *User-based* or *Item-based* CF algorithms. Unlike these works, in [27], a *SlopeOne* [53] algorithm is used to find the group members' ratings. *Hybrid* models are used to a lesser extent; most of these works [22, 44, 46, 73, 77] combine *CF* with *CB* algorithms.

GRecSys are used for recommending (i) points of interest (POI) and (ii) items. The first type of *GRecSys* is popular in the tourist domain [6, 20, 30, 35, 38, 39, 54, 55, 75, 113]. The second type of *GRecSys* uses the group members' preferences towards items to evaluate how these preferences change when interacting with other users inside a group. A wide range of solutions exist to aggregate individual preferences in a group preference [18, 23, 29, 57, 70, 109, 115]. In this work, we focus on *GRecSys* to recommend items.

One general issue in these works is the lack of consensus on how to evaluate *GRecSys* depending on the recommendation scenario. Some works use small groups between two and five members [12, 23, 79, 114], in other cases, the number of members reaches hundreds of users [2, 32, 77]. In more extreme cases as in [61] groups can have thousands of members. This lack of standard evaluation scenarios is also reflected in the way of selecting group members [16, 64, 103]. Some works use random methods, while others combine groups with specific users' characteristics [23, 79, 94]. Besides, most of these works in their evaluation do not contemplate the use of items with different properties. For example, they do not consider items with subjective attributes such as wines and items with objective attributes, like digital cameras. Therefore, the recommendations system's capacity to work for different types of items cannot be properly evaluated.

2.3 Terminology definitions for *RecSys*

In this section, we define the different individual recommender system methods that we are going to use as foundations to build *GRecSys*.

2.3.1 Collaborative Filtering (CF) RecSys. This category of *RecSys* is the most commonly used. This work studies different types of CF algorithms which can be memory-based or model-based.

Memory-based CF. The first type of CF is called memory-based CF. *RecSys* from this type use ratings to calculate the similarity between all pairs of users or items. The most commonly used are User-based CF (UBCF), and Item-based CF (IBCF). These CF approaches can also be combined in a single approach, which uses a weighted mean for the predictions of both UBCF and IBCF methods. In this work, we refer to this combined method as IUCF.

We define these approaches as follows:

•*User-based CF (UBCF).*

This approach recommends items based on the similarity of a user with other users. The first task is to find the K-nearest neighbors to the user of interest using a similarity metric. The most widely used metric is the cosine similarity between users considering their item ratings. This similarity is defined as follows [100]:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

where u and v are a pair of users, I_{uv} are the items rated in common, r_{ui} and r_{vi} are the ratings given by user u and v respectively to the item i.

The second task is to predict the rating that the user of interest would give to all items consumed by the K neighbors, but this user has not. The predicted rating is calculated as [100]:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u,v)}$$

where $sim(u, v)$ is the cosine similarity between these users, μ_u and μ_v are the mean of user u and user v ratings, $N_i^k(u)$ is the set of K neighbors of user u.

•*Item-based CF (IBCF).*

This approach focuses on what unrated items are more similar to what the user has previously consumed. This approach also has two main tasks. First, it calculates the similarity between the items. As in *UBCF*, cosine similarity is the most common metric, given as [100]:

$$sim(i, j) = \frac{\sum_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{uj}^2}}$$

where i and j are a pair of items, U_{ij} is the set of users who have rated both items, r_{ui} and r_{uj} are the ratings given by user u to the item i and j respectively.

The second task is to predict the rating that the user of interest would give to an item based on the item's K neighbors. The following equation gives this prediction [100].

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \text{sim}(i, j) \cdot \left(r_{uj} - \mu_j\right)}{\sum_{j \in N_u^k(i)} \text{sim}(i, j)}$$

where $sim(i, j)$ is the cosine similarity between items i and j, μ_i and μ_j are the mean of item i's and j's ratings, $N_u^k(i)$ is the set of K neighbors for item i.•*Combined CF (IUCF).*

This approach uses a weighted mean of predictions from the previous two approaches to predict the missing rating as follows:

$$\hat{r}_{ui} = \alpha \cdot UBCF(r_{ui}) + (1 - \alpha) \cdot IBCF(r_{ui})$$

where $UBCF(r_{u,i})$ is the predicted rating using the UBCF. Similarly $IBCF(r_{u,i})$ is the predicted rating using the IBCF, and α is a weighting factor.

Model-based CF. These methods are also called *Matrix Factorization-based* algorithms. These models use a singular value decomposition (SVD) method [49] to decompose a sparse matrix into two latent factor matrices. The first matrix is the user matrix, which indicates the user's preference for each of the latent factors. The second matrix is the item matrix, containing the weight of an item for each of the latent factors. The inner product of these two matrices is then used to predict missing ratings.

•*SVD++ (SVD).*

SVD++ [107] is an extension from the SVD algorithm. The prediction of unknown ratings is given by:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

where μ is the overall average ratings, b_u and b_i are the bias for the user and item, respectively. q_i is the item latent factors, p_u is the user latent factors, I_u is the number of items rated by the user u, and y_j is a factor vector for each item.

2.3.2 Content-based RecSys. This type of *RecSys* uses the user's profile and item's attributes to make recommendations. Usually, the metadata information from each item is used to extract its attributes. Different attributes such as item categories, genres, aspect terms in user reviews could be used. For example, for the Digital Music (MUSIC) dataset, there are 214 categories (e.g., CDs & Vinyl, Alternative Rock, Blues, Jazz). Similarly, for the MovieLens (MOVIE) dataset, items (i.e., movies) can belong to 19 genres (i.e., categories) such as Drama, Action, and Comedy. Descriptions of these datasets are presented in Section 4.1.

To build a CB recommender system, first, an item profile matrix ($IPM \in \mathfrak{R}^{I \times A}$) is generated using an importance score for each item's attributes. Where I is the set of items, and A is the set of attributes. The most commonly used score is *TF-IDF* applied to the textual description of attributes. Second, with the set of users U, a user preference matrix ($PREF \in \mathfrak{R}^{U \times I}$) is generated, having all users' preferences for all items. Third, a user profile matrix ($UPM \in \mathfrak{R}^{U \times A}$) is created to keep the user's preference for each item's attributes. The UPM is given as follows:

$$UPM = PREF \odot IPM$$

where $\odot$ is the dot-product between PREF and IPM. Using a similarity metric between UPM and IPM, we estimate the degree to which a user prefers each item. Finally, the rating predictions can be calculated as follows:

$$\hat{r}_{ui} = \mu_u + p_{ui} \cdot \mu_i$$

where μ_u is the average rating of user u, μ_i is the average rating of item i, and p_{ui} is the preference of user u for item i.

2.3.3 Hybrid RecSys. A Hybrid recommender system combines both CB and CF recommenders in a single recommender to achieve better results. Several approaches, such as mixed, switched, feature combination, and weighted, exist to form a hybrid recommender system [14]. The weighted approach is one of the most widely used. It combines the predicted ratings obtained from individual CB and CF into a weighted mean.

We define the predicted rating using a hybrid approach as follows:

$$\hat{r}_{ui} = \alpha \cdot CF(r_{ui}) + (1 - \alpha) \cdot CB(r_{ui})$$

where $CF(r_{ui})$ and $CB(r_{ui})$ are the predictions for the missing rating r_{ui} using a CF and a CB approach respectively. α is a weight parameter.

2.3.4 Neural Collaborative Filtering (NCF) RecSys. NCF is a state-of-the-art neural CF model and is considered as a generalization of Matrix Factorization models [37]. NCF approaches use a DNN architecture to combine user and item embeddings to capture their interactions directly from data. The predicted rating using NCF is expressed as:

$$\hat{r}_{ui} = a_{out}(h^\top (p_u \odot q_i))$$

where a_{out} is the activation function, h is the weights of the output layer, p_u and q_i are the latent vector for user u and item i respectively.

3 EVALUATION SETTINGS FOR GRECSYS

In this work, we are interested in knowing how *GRecSys* perform in different scenarios (i.e., group size, group type, and type of the items to recommend). Therefore, we analyze the *experimental setup* from the papers we examined, mentioned in Section 2, to determine how these studies evaluate their proposed methods in different scenarios.

Only **45.8%** of the papers mention experiments using at least one of these criteria, and in some works, the information is not complete because they were studies using real users. The following sections describe the results obtained from analyzing these papers.

3.1 Group formation

Both the number of users and the cohesion of users affect the group formation.

3.1.1 Group sizes. Pelaez et al. [71] found that in e-commerce platforms, the size of groups can affect the purchasing task. Their study concludes that larger groups have more challenges with group coordination for time to task completion than smaller groups. However, larger groups obtain moderately higher gains on average than smaller groups.

Accordingly, we analyze the literature to see how these studies handle the size of their test groups. There is not a standard about how to categorize a group size. For example, in tourism, a group size up to 10 people is considered a small group[2]; for academic instruction, small groups should be no more than five students [98], while in other domains, small group size is between 7 and 12 participants[3]. Therefore, after analyzing the group size frequencies in the different studies, we categorize the groups by their size as follows:

- Small group (S): formed by 2 to 6 members.
- Medium group (M): formed by 7 to 20 members.
- Large group (L): formed by 21 to 50 members.
- Very large group (VL): formed by more than 50 members.

Figure 3a shows the distribution of the analyzed literature using our group size categorization. From this figure, we can see that most of the existing works use small groups and medium-sized groups in their test. Only a small amount of works mention that they test their solution with groups that are larger than 20 members. We created the category VL because three documents performed their tests using hundreds and even thousands of members in their groups.

3.1.2 Group types. Given the lack of existing datasets with information about individual user preferences and group preferences for items, there is a need to form synthetic groups to test the solutions [25]. The way of creating groups for testing could affect the results. Studies have shown that group cohesion is a crucial aspect of group dynamics [28]. In a more cohesive group, a group member will be more concerned about the opinions of other members and be willing to modify her opinions if she feels attached to the group dynamics. These social and motivational forces that exist between group members are called group (or social) cohesion [10].

We analyze literature to see how these studies test their solutions using groups with different types of cohesion. Of all the studies, only [5] mentions that traditional ways of generating groups are not realistic. Therefore, it proposes a new way to form groups where some group members have similar preferences while other group members have different preferences. Some similar work is done in [68, 85], where groups are formed using a mid-range pair-wise similarity between users or using those users that are closer to a cluster centroid.

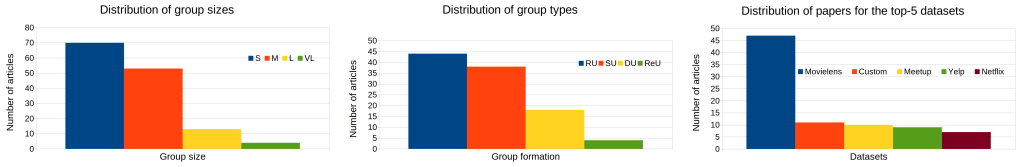
We identify four general types of group formation:

- Random (RU): a group is formed by randomly selecting users.
- Similar (SU): a group is formed with users presenting high similarity values.
- Dissimilar (DU): a group is formed with users presenting low similarity values.
- Realistic (ReU): a group is formed with users having similar and dissimilar preferences.

The first three ways of group formation are the most used in the studies. However, variations on the way of forming the groups exist. For example, some works first calculate the average similarity for the whole group and then find which users are more similar or dissimilar to the average, while others use a pairwise similarity value to get the next user for a group.

Figure 3b shows the distribution of papers using our group type categorization. We can observe the different approaches used by the studies. The use of random groups for testing is the most applied method, and it could form groups with users having a high similarity or a high dissimilarity. For some activities, random groups can be formed. However, members of these groups shared

[2]https://arival.travel/whats-the-optimal-tour-group-size/
[3]https://topgolf.com/us/plan-an-event/



(a) Group Sizes (b) Group Types (c) Datasets

Fig. 3. Distribution of papers mentioning a) group sizes, b) group types, and c) distribution of top-5 datasets.

preferences in different degrees for a given task. Therefore, in some cases, it is preferred to have groups with totally dissimilar members. For example, testing a new product before launching it to production is desirable to know a broader set of opinions. Therefore, it is of great importance to test the performance of *GRecSys* with these types of groups.

3.2 Item types

There is the premise that better algorithms lead to better recommendations, which leads to better user satisfaction and perceived system effectiveness [48]. However, users' experience in e-commerce environments is also affected according to the types of items on which they collect information and make a purchase decision [41, 52]. Items that can be bought in e-commerce websites can be classified as [41]:

- Search goods (SG): their perceived quality involves objective attributes, that can be discovered before purchasing and without interacting with the item. For example, digital cameras or clothes.
- Experience goods (EG): their perceived quality depends more on subjective attributes that are a matter of personal taste, which only can be discovered through the evaluation of the product. For example, music, wine, or movies.

It follows that a purchase decision for *SG* may have different information requirements than a purchase decision for *EG*. Existing works on *GRecSys* do not make a detailed evaluation of how they behave when recommending different types of goods. Most of these works compare their proposals and baselines using datasets of *EG*, for example, movies, music, beer, or venues (i.e., POI). In the literature, we are only aware of one article [102] that uses *SG*. However, the paper does not clearly define what these items are.

Figure 3c shows the distribution of articles for the top-5 datasets used in our reviewed papers. We grouped those papers using any version of the MovieLens dataset[4]. This dataset is the most widely used for testing *GRecSys*, and the items belong to the *EG* category. The second frequently used datasets are specifically designed to test *GRecSys* in studies involving real users. These datasets have specific qualities for each study and are not publicly accessible. The items that these datasets recommend are songs, movies, or locations. Therefore they fall into the category of *EG*. The third and fourth frequently used datasets are used in studies recommending POI, like restaurants (i.e., Yelp[5], Meetup[6]). These datasets also belong to *EG*. Finally, the fifth most used dataset is the Netflix dataset[7], which is similar to the MovieLens dataset and belongs to *EG*.

In conclusion, we found that **99.4%** of the studies related to *GRecSys* evaluate their approaches using items from the *EG* category. This number indicates that it is necessary to evaluate *GRecSys* with other types of datasets to avoid the bias that may exist when using only a particular item type.

[4]http://grouplens.org/datasets/movielens/
[5]https://www.yelp.com/dataset/challenge
[6]https://www.meetup.com/es/topics/open-data/
[7]https://www.kaggle.com/netflix-inc/netflix-prize-data

Table 2. Description of the Amazon and MovieLens datasets.

Type	Dataset	# users	# items	# reviews	avg. rating	std.
SG	TOOLS	16,638	10,217	134,476	4.36	1.03
	OFFICE	4,905	2,420	53,258	4.34	0.93
	GARDEN	1,686	962	13,272	4.19	1.08
EG	FOOD	14,681	8,713	151,254	4.24	1.09
	MUSIC	5,541	3,568	64,706	4.22	1.09
	INST	1,429	900	10,261	4.49	0.89
	MOVIE	**943**	**1,682**	**100,000**	**3.53**	**1.12**

4 EXPERIMENTAL SETUP

This section presents the different elements used to test *GRecSys*. For this study, we are only interested in the recommendation of items. Therefore, we do not analyze works for POI recommendation. Section 4.1 explains the characteristics of the different **datasets**. Section 4.2 presents the implementation details for each **base recommender system model**. Section 4.3 describes the approach taken to generate **groups** based on sizes and types. Sections 4.4 and 4.5 describe the implementation of the **aggregation strategies** and **aggregation functions** respectively. Finally, Section 4.6 presents the **metrics** to evaluate the methods and the way they are calculated.

4.1 Datasets

We want to test if the type of items (i.e., EG and SG) influences the *GRecSys* results. Datasets for testing *GRecSys* are usually obtained from datasets used for testing *RecSys* for individual users [8, 34] due to the lack of publicly available datasets with ground truth information for *GRecSys* [25, 81]. For testing *RecSys*, EG datasets are the more often used type and the most widely used dataset is MovieLens-100k (**MOVIE**). This work uses such datasets. Furthermore, we argue that it is important to test *GRecSys* using other real-life datasets by analyzing other datasets' characteristics.

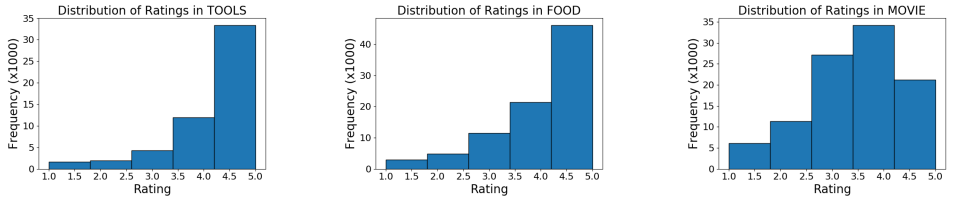
We test different *GRecSys* using real-life datasets for both types of items (i.e., EG and SG). The selected datasets come from the Amazon Review dataset [36]. It contains reviews and metadata for different types of items spanning from May-1996 to July-2014. Table 2 shows the description of all datasets used in this work. The datasets include Tools and Home Improvement (**TOOLS**), Office Products (**OFFICE**), Patio Lawn and Garden (**GARDEN**), Grocery and Gourmet Food (**FOOD**), Digital Music (**MUSIC**), and Musical Instruments (**INST**).

One characteristic of these real-life datasets is their sparsity compared with the MOVIE dataset. Sparsity is defined as [4]:

$$sparsity = (1 - \frac{|reviews\_in\_dataset|}{|users| \times |items|}) \times 100 \quad (1)$$

where $|reviews\_in\_dataset|$ is the total number of reviews in the dataset, $|users|$ and $|items|$ are the numbers of users and items, respectively.

A low number of ratings causes high sparsity. Table 3 shows that the MOVIE dataset is a denser dataset, where on average, users have rated many more items than those in the Amazon datasets. From this, we can see that real-life users rate a small number of items, as stated in the literature [43], whereas the MOVIE dataset may not reflect how real-life users rate items. For example, MovieLens asks users to rate at least 15 items on the sign-up process [34].



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 4. Dataset ratings distribution.

Table 3. Comparing Amazon datasets with MovieLens (MOVIE) dataset

Dataset	Sparsity	Ratings per user	Ratings per item	Missing Ratings per user
TOOLS	99.9%	8.08	13.16	≈ 10,208
OFFICE	99.5%	10.86	22	≈ 2,409
GARDEN	99.1%	7.87	13.79	≈ 954
FOOD	99.8%	10.3	17.35	≈ 8,703
MUSIC	99.6%	11.67	18.13	≈ 3,556
INST	99.2%	7.18	11.4	≈ 893
MOVIE	**93.6%**	**106.04**	**59.45**	**≈ 1,576**

Another difference between the MOVIE dataset and the Amazon datasets is in rating distribution. As Figure 4 shows, users in real life tend to give high ratings to items, whereas in the MOVIE dataset, we can observe more diverse distribution among the rating range.

These findings (i.e., rating sparsity and distribution) motivate the importance of testing *GRecSys* with real-life datasets. For our tests, we remove from the Amazon datasets those users who have rated less than ten items.

4.2 RecSys setup

4.2.1 Collaborative Filtering. Four *RecSys*, *UBCF*, *IBCF*, *IUCF*, and *SVD* are utilized. All methods are implemented using the Surprise[8] framework. In particular, the first three methods use the basic *KNNWithMeans* CF algorithm implementation by setting K to be 50 and considering the mean ratings of each user. For *IUCF*, we set the rating weight of *UBCF* to be 0.6 and that of *IBCF* to be 0.4. For SVD, where the framework uses a stochastic gradient descent (SGD) approach, we set the number of factors to be 20, and the number of iterations of SGD method to be 20 with a learning rate of 0.1.

4.2.2 Content-Based. To the best of our knowledge, no existing framework fully implements a CB model. We build a CB recommender system using the items categories and subcategories as attributes for the Amazon datasets. For the MOVIE dataset, we use the genres as attributes. Table 4 shows the number of categories (i.e., attributes in content based recommender systems) of each dataset.

The PREF matrix contains values 1 or 0 depending on whether the user rated the item or not. We use the cosine similarity between the user profile and the item profile.

[8]https://surprise.readthedocs.io/en/stable/

Table 4. Number of categories of each dataset

	TOOLS	OFFICE	GARDEN	FOOD	MUSIC	INST	**MOVIE**
Categories (attributes)	1,266	793	842	512	214	809	**19**

4.2.3 Hybrid. We combine the *SVD* method (as a CF approach, with rating weight 0.6) and the CB method (with rating weight 0.4). We give a higher weight to SVD given that CB algorithms suffer from overspecialization [97].

4.2.4 NCF. To implement this approach, we use the *fast.ai*[9] framework for collaborative filtering. As hyper-parameters, we use a learning rate of 0.01. The number of factors is set to 20 to be aligned with SVD. We train the model for 5 epochs to avoid overfitting.

4.3 Group formation

An essential part of this study is group formation. According to the literature, different types of group formation have been used. We utilize approaches in different *GRecSys* [15, 80, 83, 90, 109] to combine group types and group sizes. For all the tests, we form 100 different groups. The following sections describe the procedure to form these groups.

4.3.1 Group size. We use four different group sizes (details see Section 3.1.1): S(mall), M(edium), L(arge), and VL (Very Large). This study analyzes *GRecSys* performance for different group sizes in general, but not how *GRecSys* behave when the group size monotonically increases. We are not interested in differences of *GRecSys* performance for a group of three members and a group of four members.

4.3.2 Group type. We define four group types (details see Section 3.1.2): RU (Random), SU (Similar), DU (Dissimilar), and ReU (Realistic). The vast majority of past works use RU to form their groups. For the other types, the similarity between users is used to select users in a given group.

4.3.3 Group creation. To form a group of size K, when the type of group is RU, we randomly select K users to form the group. For the other group types (i.e., SU, DU, ReU), we first split all users into two clusters. This step allows us to exploit the cluster properties, where members of each cluster present a high intra-similarity within their cluster and low inter-similarity with other cluster members.

For the SU type, we randomly select a user and get her cluster. We calculate this user's cosine similarity with the rest of the users of the same cluster and select the K-1 most similar users to this user to form the group.

For the DU type, we also randomly select a user and get her cluster. We calculate this user's cosine similarity with users of the other cluster. The users from the other cluster must initially be dissimilar to the selected user, even though, using the similarity measure, we select the K-1 *least* similar users to this user to complete the group.

Finally, for the ReU type, we want to simulate a more realistic scenario, where inside a group, there could be members similar to others, and at the same time, users dissimilar to others. To form the groups, we combine the two previous procedures (i.e., SU and DU). First, we randomly select a user. Then, we select her most similar users following the procedure for the SU type, and her most dissimilar user follows the steps for the DU type. Then, we merge both sets of users (i.e., SU and DU), and randomly select without replacement K-1 users to complete the group.

[9]https://docs.fast.ai/collab.html

4.4 Aggregation strategies

We follow the two aggregation strategies presented in Section 2.2.1. For the *aggregated predictions* (PRED) strategy, for each group, we first get the set of items not rated in common by its group members. Then, depending on the selected recommender system, we predict the missing rating of each item for each group member. Next, once we have the predictions for all group members, we apply the selected aggregation function (Section 4.5). Finally, we sort the results from the previous step in descending order, and we get as the group recommendation the top-N higher rated items.

For the *aggregated models* (PROF) strategy, we need to create a profile for a *"virtual"* user to represent each group. First, for each group, we obtain the initial item ratings for each of its group members. Second, we aggregate the group members' ratings using the selected aggregation function (Section 4.5). The result of this step forms the preference profile for the *"virtual"* user representing the group. Third, depending on the selected recommender system, for each *"virtual"* user, we predict its items' missing ratings. Finally, we get as the group recommendation the top-N higher rated items. For both strategies, we select the top 100 items as the group recommendation list.

4.5 Aggregation functions

Based on [26], we implement ten aggregation functions (see Section 2.2.2) for both aggregation strategies. We denote the set of users as U, a group of users as $G \subset U$, a group member as $u \in G$, the set of items rated by G as I, and item $i \in I$.

4.5.1 Additive (ADD). This function ranks items based on the addition of group members' ratings. It is defined as:

$$\underset{(i \in I)}{\text{argmax}} \left(\Sigma_{u \in G} \, \text{rating}(u, i)\right)$$

4.5.2 Approval (APP). For each item, this function counts how many group members have rated it above a threshold. Then, items are ranked using this count.

$$\underset{(i \in I)}{\text{argmax}}(\mid \{u \in G : \text{rating}(u, i) \geq \text{threshold}\} \mid)$$

For both strategies, we use a threshold of 3.0 (we consider that items with ratings close to 4 or 5 are more to the liking of users).

4.5.3 Average (AVG). In this function, for each item, the average of group members' ratings is calculated, then the items are ranked based on their average.

$$\underset{(i \in I)}{\text{argmax}} \left(\frac{\Sigma_{u \in G} \, \text{rating}(u, i)}{|G|}\right)$$

4.5.4 Average Without Misery (AWM). In this function, items with ratings below a certain threshold are removed from the candidate list, then items are ranked based on their average.

$$\underset{(i \in I, \nexists u \in G | rating(u,i) \leq threshold}{\text{argmax}} \left(\frac{\sum_{u \in G} \text{rating}(u, i)}{|G|}\right)$$

For both strategies, we use a threshold of 2.0 (items with low ratings are not of users' interest).

4.5.5 Borda Count (BC). This function first assigns a score to the items based on their ranking in the list of each user. Then, it ranks the items using their total score.

$$\underset{(i \in I)}{\text{argmax}} \left(\Sigma_{u \in G} \, \text{score}(rating(u, i))\right)$$

4.5.6 Least Misery (LM). This function recommends the item that maximizes the minimum rating of all the group ratings.

$$\underset{(i \in I)}{\text{argmax}}(\min_{u \in G}(\text{rating}(u, i)))$$

4.5.7 Most Pleasure (MP). This function measures the group's satisfaction by using the maximum group members' ratings. I.e.,

$$\underset{(i \in I)}{\text{argmax}}(\max_{u \in G}(\text{rating}(u, i)))$$

4.5.8 Most Respected Person (MRP). This function considers the expertise of group members towards an item. The ratings of the users who are considered experts are used for the group.

$$\underset{(i \in I)}{\text{argmax}}\ (\text{rating}\ (u', i))$$

where $u' \in G$ is the most respected user.

4.5.9 Multiplicative (MUL). In this function, items are ranked based on the result of multiplying group members' ratings.

$$\underset{(i \in I)}{\text{argmax}}\ (\Pi_{u \in G}\ \text{rating}(u, i))$$

4.5.10 Popularity (POP). This function considers those items which are the most popular.

$$\underset{(i \in I)}{\text{argmax}}(\text{count\_ratings}(i))$$

We use a threshold of 20 ratings considering the average of ratings per item for the datasets.

4.6 Metrics

When recommending items to users, it is essential to consider different performance metrics. We use both quantitative and qualitative evaluations to evaluate the results from the aggregation strategies and functions applied to each recommender system approach. We calculate these metrics for each group and report as the final value the median of each metric.

4.6.1 Quantitative metrics.

Hit Ratio. This metric calculates the average percentage of items in the group recommendation list present in the individual member recommendation list [25].

$$HR_{Group} = \frac{1}{|G|} \sum_{u=1}^{|G|} \frac{|R_u \cap R_{Group}|}{|R_{Group}|} \tag{2}$$

Here, R_u and R_{Group} are the recommendation lists for each group member and for the group respectively.

nDCG@k. We adopt the Normalize Discounted Cumulative Gain (nDCG) metric [104] to measure the ranking quality of each group's list of recommended items. This metric is the most popular among all the works analyzed. It consists of the Discounted Cumulative Gain (DCG) and the Ideal Discounted Cumulative Gain (IDCG) defined below.

$$DCG_k = \sum_{i=1}^{k} \frac{rating(i)}{log_2(i+1)},\ IDCG_k = \sum_{i=1}^{rel} \frac{rating(i)}{log_2(i+1)}$$

Table 5. Summary of elements' acronyms

Component	Element Acronyms
Datasets	TOOLS, OFFICE, GARDEN, FOOD, MUSIC, INST, MOVIE
Item types	SG, EG
Group sizes	S, M, L, VL
Group types	RU, SU, DU, ReU
Recommeder Systems	UBCF, IBCF, IUCF, SVD, CB, HYBRID, NCF
Aggregation strategies	PRED, PROF
Aggregation functions	ADD, APP, AVG, AWM, BC, LM, MP, MRP, MUL, POP
Metrics	HR, nDCG@5, Diversity, Coverage

where $rating(i)$ is the rating for an item in the i-th position, and rel ($\leq k$) is the number of items with the highest ratings up to a position k in the recommendation list. Then

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{3}$$

4.6.2 Qualitative metrics.

Diversity. Diversity measures how different the items are in a recommendation list. Having diversification in the recommendation list increases the quality of the user's experience because it reduces the recommender system over-specialization [50].

First, we calculate the Intra-List Similarity (ILS) [50] for each group recommendation $ILS_{Group} = \frac{1}{2}\sum_{i\in \mathrm{L}}\sum_{j\in L}\mathrm{sim}\,(i,j)$, where L is the list of recommended items, and $sim(i,j)$ is the cosine similarity between items i and j. To calculate the similarity, we use the items' categories for the Amazon datasets and the MOVIE genres. Second, we calculate the final group's list diversity [50] as follows:

$$diversity_{\mathrm{Group}} = 1 - ILS_{Group} \tag{4}$$

Coverage. This metric represents the percentage of items in the recommendation list that a recommender system can recommend among the number of potential items (i.e., catalog). A higher coverage may benefit users by exposing them to a broader range of recommended items, which could increase satisfaction with the recommender system [1].

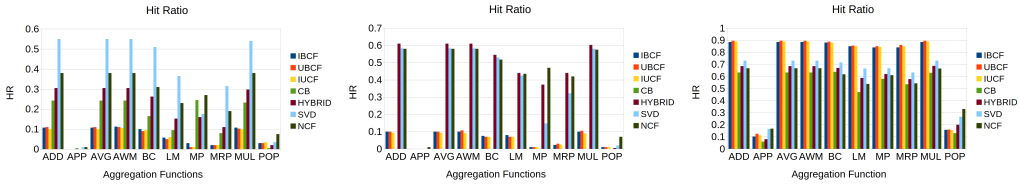
First, we form the item catalog for the group using the items from all individual group member's recommendation list [1] as $catalog_{Group} = \bigcup_{u=1}^{|G|} R_u$ where R_u is a group member's recommendation list. Then, we calculate the catalog coverage for the group by getting the ratio of items in the group recommendation list R_G that exist in the group catalog as follows [1]:

$$CatalogCoverage_{Group} = \frac{|R_G \cap catalog_{Group}|}{|catalog_{Group}|} \tag{5}$$

To end this section, we present Table 5, which shows the elements' acronyms for the different defined components we are going to use for our evaluations. We present this table to make it easier for the reader to read the results in the next section. All these elements are defined above.

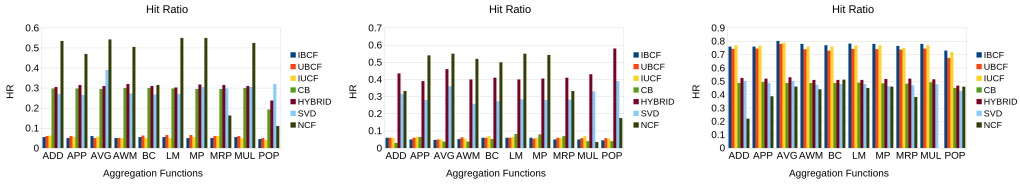
5 PERFORMANCE EVALUATION

Several works show that no single recommender system approach is better than others for all scenarios [17]. We only show results from two Amazon datasets (i.e., TOOLS and FOOD) because these datasets have the most significant number of reviews. These datasets represent a real-life environment where many users have only rated a small percentage of items. Recall that TOOLS and FOOD are SG and EG, respectively. The results from the other Amazon datasets show similar



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 5. HR metric for all recommender systems using PRED aggregation strategy



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 6. HR metric for all recommender systems using PROF aggregation strategy

conclusions and are omitted here due to space limitation. All the results can be found in the Appendix. We also present the results for MOVIE as a comparison baseline.

5.1 Effect of different aggregation functions and aggregation strategies

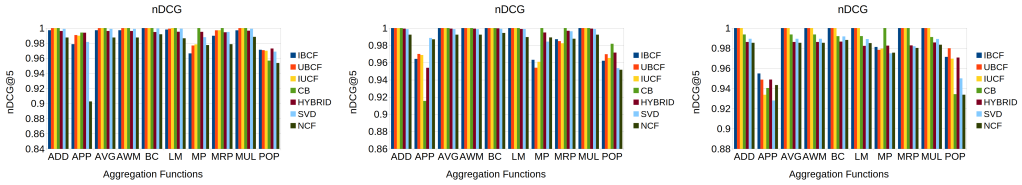
In this section, we analyze how aggregation functions influence the group recommendations when different aggregation strategies are used. We fix the group size to *S(mall)* and the group type to *RU (Random)* for this set of tests.

5.1.1 Hit Ratio. Figures 5 and 6 show the HR results for the TOOLS, FOOD, and MOVIE datasets using both aggregation strategies (PRED and PROF)).

Figure 5 shows the HR results when the aggregation strategy is PRED. On the one hand, for SG items, Figure 5a indicates that the RecSys producing the best HR results is SVD in combination with ADD, APP, AVG, AWM, BC, LM, MRP, and MUL. If the MP or POP functions are used, NCF is the best approach, although the HR values are considerably lower than others. UBCF, IBCF, and IUCF have the worst performance. This result is possible because these methods are based on the similarity between neighbors. Given the sparsity of the dataset, their performance is affected.

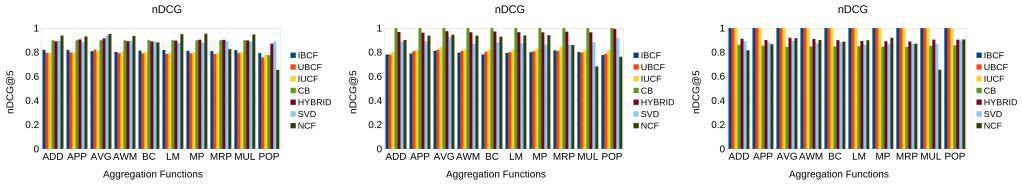
On the other hand, when items are from EG, Figure 5b shows that the approach with the best results is HYBRID when it is combined with ADD, AVG, AVM, CB, LM, MRP, MUL. Again, if MP or POP functions are used, NCF is the best approach. For MOVIE, Figure 5c shows that the best approaches for most of the aggregation functions are UBCF, IBCF, and IUCF. These results make sense if we account that MOVIE is less sparse than TOOLS and FOOD, indicating that each user and each item has more similar neighbors with similar ratings.

Figure 6 shows the results for HR when the aggregation strategy is PROF. When this aggregation strategy is used, we can observe from Figures 6a and 6b that overall the best recommender is NCF in combination with aggregation functions APP, AVG, AWM, BC, LM. For SG (i.e., TOOLS), a proper combination is with ADD and MUL; however, with POP aggregation, the best approach is SVD. For EG (i.e., FOOD), using ADD, MRP, MUL, and POP, the best recommender approach is HYBRID. Again, the behavior of UBCF, IBCF, and IUCF is similar to the use of the PRED aggregation strategy.



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 7. nDCG@5 metric for all recommender systems using PRED aggregation strategy



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 8. nDCG@5 metric for all recommender systems using PROF aggregation strategy

Overall, the highest HR values are obtained when using the PRED aggregation strategy, when the aggregation is done at the end using the predicted ratings. These results indicate that when looking for high HR values, the aggregation function is influenced by different factors such as the aggregation strategy, the type of items, the sparsity of the dataset, and the number of neighbors for each user and item.

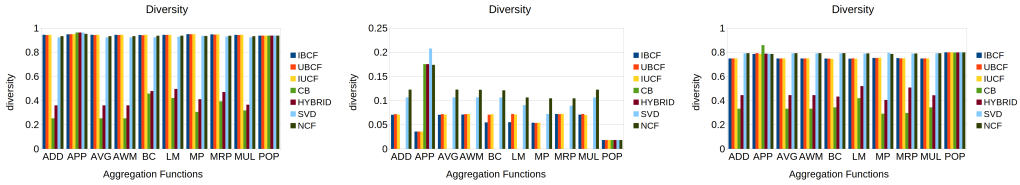
5.1.2 nDCG@k. The HR results measure the degree that the items recommended for the group match those recommended items for each group member. In this section, we analyze to what degree the ranking of items for the group recommendation matches the ranking of items for each group member. These results are shown in Figures 7 and 8.

Figure 7 presents the result of the PRED aggregation strategy. Figures 7a and 7b show that for both types of items (i.e., SG and EG), most methods get similarly good results with most of the aggregation functions. Only when using APP, MP, or POP aggregation functions most of the recommender approaches get low values. On the other hand, Figures 8a and 8b show that for both SG and EG items, CB, HYBRID, SVD, and NCF have the best results when using the PROF aggregation strategy combined with most of the aggregation functions. While, IBCF, UBCF, and IUCF show the lower results.

Figures 7c and 8c show that for MOVIE, in general, the best recommender systems are IBCF, UBCF, and IUCF using any aggregation function. When comparing these results, overall, the highest values in nDCG are obtained when using the PRED aggregation strategy.

These results indicate that the distribution of ratings (see Figure 4) in the dataset is an important factor for the high nDCG values for those approaches. As the distribution of ratings for the real-life datasets is skewed toward the highest ratings (i.e., 4 and mainly 5), then it is expected that the ratings in the first places of the recommendation list are high. These results also indicate that when looking for high nDCG values, the aggregation function is influenced by different factors such as the aggregation strategy and the rating distribution.

We have analyzed the quantitative performance of different *GRecSys* on different item types. Besides, we analyzed their performance when different *RecSys* are used in combination with two aggregation strategies and ten aggregation functions.



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 9. Diversity metric for all recommender systems using PRED aggregation strategy

In the next sections, we analyze *GRecSys* qualitative performance.

5.1.3 Diversity. Diversity measures how different the items are that are recommended to the group. Figure 9 shows the results of the PRED aggregation strategy. Comparing Figures 9a and 9b, we can see that the FOOD dataset's diversity values are much lower than those for TOOLS. This result indicates that the recommended list contains very similar items in the FOOD dataset. We attribute this behavior to two aspects. First, items in FOOD are not distributed among many categories like TOOLS (see Table 4). Second, the number of items with similar ratings in FOOD is higher than in the TOOLS dataset (see Figure 4). This behavior is not present in other datasets (e.g., OFFICE, MUSIC, GARDEN, INST) where items are distributed among more categories.

Figure 9a shows that for the TOOLS dataset, IBCF with ADD, AVG, LM, MP, and MRP recommends more diverse lists of items. This result may happen because TOOLS is a very sparse dataset, affecting the rating predictions as items do not have many neighbors, then neighbors with different characteristics are used. Similarly, IUCF presents good results using AWM, BC, and MUL. As this approach combines users and items, it works best with those functions that maximize the ratings. Figure 9b shows that NCF, with most aggregation functions, provides the best results on the FOOD dataset, even when all values are much lower. Only when using the APP aggregation function, SVD shows the best diversity score. This particular behavior of low values is not shown in the other datasets for EG items (e.g., MUSIC, INST), where NCF is still the best option.

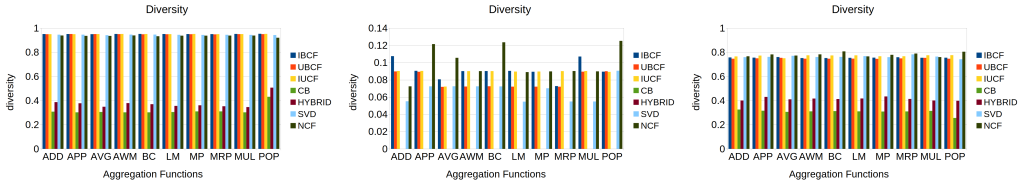
Figure 10 shows the results of the PROF aggregation strategy. We can see the same trend of low diversity values for the FOOD dataset (see Figure 10b). We attribute this behavior of low values to the same reasons mentioned for the PRED aggregation strategy.

Figure 10a shows that higher diversity values for TOOLS are obtained using IBCF with ADD, BC, LM, MRP, MUL, and POP, followed by UBCF with APP, AVG, and IUCF with AWM and MP. This result possibly appears because of the dataset sparsity, and therefore neighbors with different features have to be used. Figure 10b shows that the approach showing better results is NCF combined with APP, AVG, BC, MP, MRP, and POP. This particular behavior is not present in other datasets for EG items (e.g., MUSIC, INST). For these datasets, the best approaches are obtained using UBCF and IUCF, respectively.

Figures 9c and 10c show that the best approach is NCF when using the MOVIE dataset. Recall that ratings are well distributed in this dataset, and items belong to a small number of genres. These dataset properties could lead NCF to find better rating patterns.

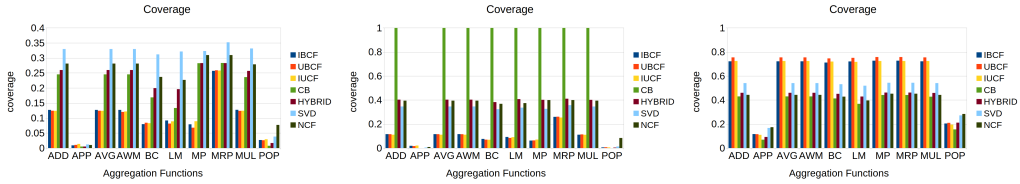
The presented results indicate that when looking for high Diversity values, the aggregation function is influenced by different factors such as the aggregation strategy and the distribution of items concerning their features (i.e., categories, genres).

5.1.4 Coverage. Figure 11 shows the results when using the PRED aggregation strategy. Figure 11a shows that the SVD approach, combined with most aggregation functions, obtains the best results. The highest values are obtained with the MRP aggregation function. On the other hand, Figure 11b shows a high Coverage for FOOD when using CB with almost all the aggregation functions. Only



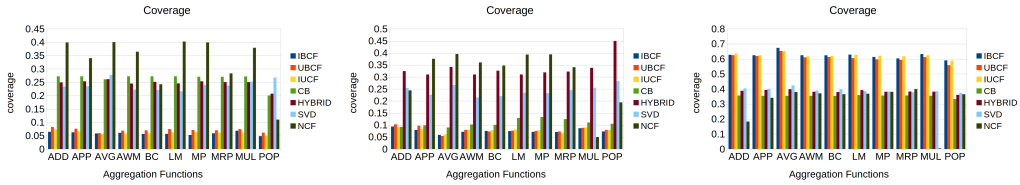
(a) TOOLS (b) FOOD (c) MOVIE

Fig. 10. Diversity metric for all recommender systems using PROF aggregation strategy



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 11. Coverage metric for all recommender systems using PRED aggregation strategy



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 12. Coverage metric for all recommender systems using PROF aggregation strategy

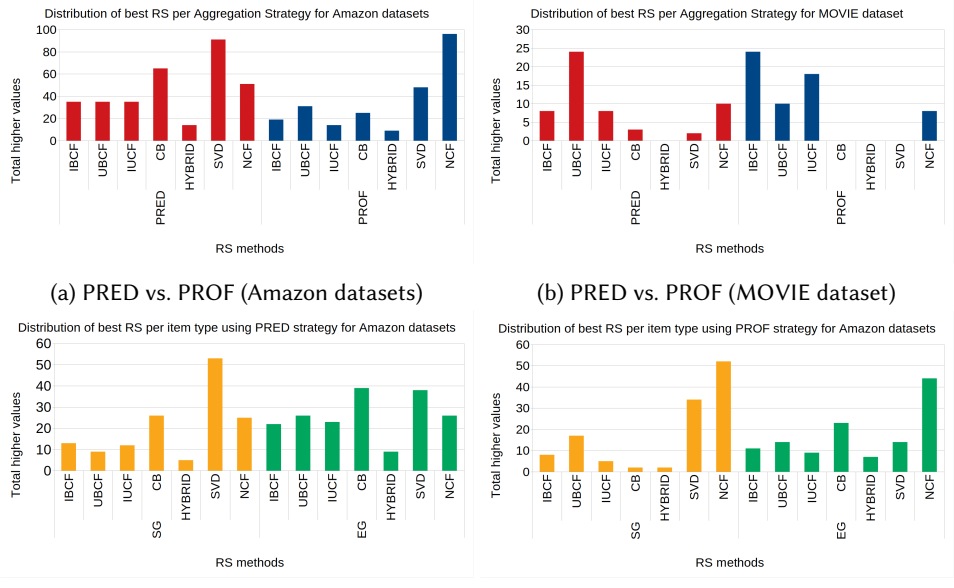
with APP and POP aggregation functions, CB's values are low. This behavior is not consistent with the other EG datasets, where SVD is the most dominant approach. As in Diversity, we attribute this behavior to the two characteristics of FOOD. Moreover, as CB recommends items based on their features (i.e., categories or genres), the recommendation list created by most of the aggregation functions covers the catalog of items.

Figure 12 shows the results when using the PROF aggregation strategy. Figure 12a shows that for the TOOLS dataset, NCF presents the best results in combination with most of the aggregation functions (ADD, APP, AVG, AWM, LM, MP, MRP). This behavior is consistent with the mid-size SG dataset OFFICE. For the smallest SG dataset, SVD is the most effective approach using most of the aggregation functions. Figure 12b shows that NCF is the best approach with most of the aggregation functions. This result is also consistent with the other EG datasets. This figure also shows that the HYBRID method has high values for ADD, MUL, and POP. HYBRID results are probably dependent on SVD, which is also high for these aggregation functions, boosted with CB given that the aggregation step happens earlier in the recommendation process.

Figures 11c and 12c show that when using the MOVIE dataset, the best approaches are UBCF, IBCF, and IUCF when using any aggregation functions. These results indicate that the influence of similar neighbors is strong in this dataset.

Results indicate that when looking for high Coverage values, the aggregation function is influenced by different factors such as the aggregation strategy, the sparsity in ratings, and the distribution of items regarding their features (i.e., categories, genres).

In this section, we analyze the use of aggregation functions with different *RecSys*. Results show that in most cases, the aggregation functions ADD, AVG, AWM, and MUL produce better results with the majority of the recommendation approaches. For each metric, we identify a set of factors that influence the results of these aggregation functions, such as the rating distribution and the aggregation strategy.



(a) PRED vs. PROF (Amazon datasets)

(b) PRED vs. PROF (MOVIE dataset)

(c) SG vs. EG (PRED)

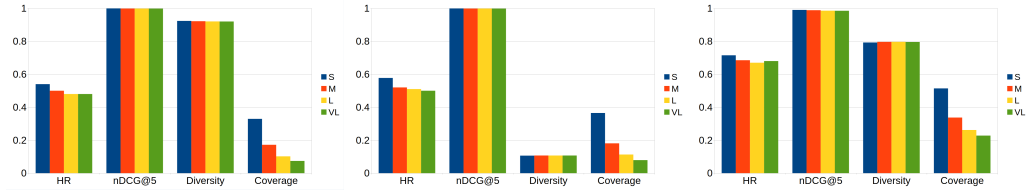
(d) SG vs. EG (PROF)

Fig. 13. Distributions of best RecSys methods for different aggregation strategies and item types

5.1.5 RecSys and aggregation function selection. This section examines which recommender system works best for a specific aggregation strategy (i.e., PRED and PROF) and a type of items (i.e., SG and EG). We count the frequencies of tests in which a recommender system method obtains the best performance.

We first examine which methods win for different aggregation strategies (PRED and PROF). Figure 13a and 13b show the results. For Figure 13a, 240 test results were collected for each aggregation strategy. They correspond to the four metrics results on sixty experiments (using the ten aggregation functions on the six Amazon datasets). For 13b, 40 test results were collected. They are the values of the four metrics on ten experiments (using the ten aggregation functions on the MOVIE dataset). When we have ties between some recommender system methods, the count for each of them is increased by one.

Figure 13a shows that for the PRED aggregation strategy, the SVD approach has the highest value, followed by CB. For the PROF aggregation strategy, the NCF approach is the one with the highest value. This result indicates that we could use SVD or NCF depending on the aggregation strategy. For the MOVIE dataset (results in Figure 13b), we can see that the best algorithm for the PRED aggregation strategy (in red) is UBCF, while for the PROF aggregation strategy (in blue), IBCF wins. This difference in algorithms between the two types of datasets (i.e., Amazon datasets



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 14. Metric results for different group sizes using SVD method and PRED aggregation strategy

and MOVIE dataset) could be attributed to the number of ratings each user has made on average, and also the number of ratings each item has on average. As Table 3 shows, while for Amazon datasets, these are low numbers, in the MOVIE dataset, these numbers are considerably higher.

We further investigate the winning methods for recommending different types of items (SE and EG). Figure 13c and 13d show the results for the SG type (in color orange) and for the EG type (in color green) respectively. For both figures, 120 tests results are collected for each type of item. These results correspond to the four metrics values collected from thirty experiments (using ten aggregation functions on the three Amazon datasets of each type). The ties are processed using the same strategy for Figures 13a and 13b.

Figure 13c shows that when the aggregation method is PRED, for the SG type, the clear winner is the SVD algorithm, whereas, for the EG type, the CB approach is slightly better than the SVD. On the other hand, Figure 13d shows that when the aggregation method is PROF, for both item types (i.e., SG (in orange) and EG (in green)), the NCF approach has the highest frequency value.

Regarding the aggregation function selection, we consider that users are willing to modify their personal preferences to reach an agreement for the benefit of the group [19]. The AVG aggregation function is similar to making decisions in a group of people [58]. Moreover, the results in previous Sections show that the AVG function has better performance than the AWM function. Therefore, this is an additional argument to choosing the AVG function as the most optimal function for use in *GRecSys*.

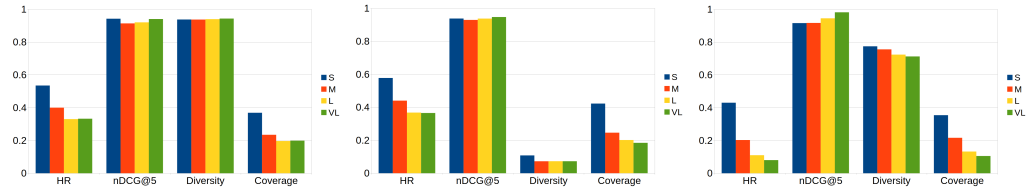
5.2 Effect of group formation

This section examines how the group formation (sizes and types) affects the performance of recommender approaches. We measure the performance using the four metrics. We use SVD and NCF methods for the PRED and PROF aggregation strategies respectively. We use AVG as the aggregation function. These are the suggested effective methods and functions as shown in analysis in Section 5.1.5

5.2.1 Effect of different group sizes. We first check the effect of group sizes. For these tests, we fix the group type as RU (Random). Figure 14 demonstrates how the group sizes affect the SVD method when the aggregation strategy is PRED.

Figure 14 shows that, on the three datasets, the best HR and Coverage scores are obtained with group size S, while the worst score is from the VL groups. For nDCG and Diversity, there is no significant differentiation in the group sizes. The results indicate that there is a better agreement in small groups than in larger ones. Similar results can be observed from tests on other SG and EG datasets (see the Appendix).

Figure 15 shows the results obtained for the PROF aggregation strategy using the NCF method. We can observe that the best HR and Coverage results are obtained with the group size S, while the lower score is from the VL groups. Regarding nDCG, on TOOLS, group size S is slightly better than VL, and for Diversity, we see a change, where VL groups obtain the best values. However, these results are not followed by the the experimental results on other SG datasets. A possible



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 15. Metric results for different group sizes using NCF method and PROF aggregation strategy

explanation could be the higher number of missing ratings in TOOLS compared with the other SG datasets.

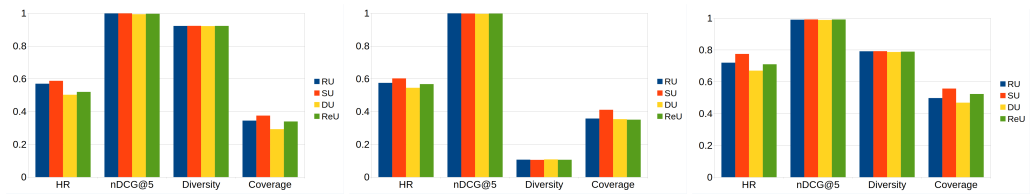
For the FOOD and MOVIE datasets, we found a reverse pattern on nDCG, where VL groups have better values than S groups. Mainly because the "virtual" user profile must cover more items, helping the NCF method find useful patterns. This pattern is consistent with the other EG datasets. For Diversity, for these two datasets, the best results are obtained with the group size S. This result is not consistent with the other EG datasets. One reason could be the difference in the rating distribution in these datasets.

When the size of the group grows, the number of individual preferences that must be considered during the recommendation process also grows. Therefore, recommending new and appealing items for all members becomes more difficult. This finding aligns well with the literature. Moreover, results using the PROF aggregation strategy (i.e., Figure 15) show more notable differences between group sizes than those using the PRED aggregation strategy (i.e., Figure 14), because of the creation of a broader profile for the group.

5.2.2 Effect of different group types. In this section, we check the effect of group types. For these tests, we fix the group size to be *S(mall)*.

Figure 16 presents the results for the PRED aggregation strategy using the SVD method. For the HR and Coverage metrics on the three datasets (i.e., TOOLS, FOOD, and MOVIE), we can see that the highest values are obtained when the group is formed by similar users (i.e., SU). Correspondingly, the lowest values are obtained when the group has dissimilar users (i.e., DU). These results make sense since the profiles of group members should be more related in SU and more diverse in DU. For both the nDCG and Diversity metrics, there is not much difference in the results for the different group types. The Diversity values for the FOOD dataset is lower than for the other datasets. This behavior is aligned with the results of Section 5.1.3, as the FOOD dataset is not distributed in many categories, and its number of items with similar ratings is higher compared with TOOLS.

These results indicate that as more like-minded the group members are (i.e. SU), the group becomes more homogeneous. Therefore, the ratings these similar users give to items should also be similar.



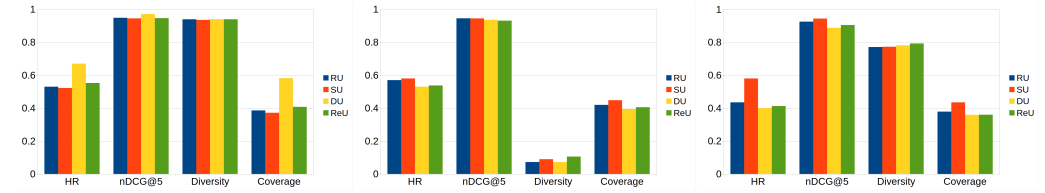
(a) TOOLS (b) FOOD (c) MOVIE

Fig. 16. Metric results for different group types using SVD method and PRED aggregation strategy

Figure 17 presents the results for the PROF aggregation strategy using the NCF method. Figure 17a shows that the DU (Dissimilar) type has the highest value for HR, nDCG, and Coverage metrics. For Diversity, there is not a clear dominant group type. Consistent results can be observed from experiments on other SG datasets (see the Appendix). One possible explanation for these outcomes is that when the profile aggregation is done at an early stage in the group recommendation process, the *"virtual"* user profile is broader, which can be similar to more users in the dataset.

Figure 17b shows that the pattern of the results is similar to the results for the MOVIE dataset (Figure 17c). The Diversity for values for this dataset is lower than for the other datasets. This behavior is aligned with the results of Section 5.1.3.

Overall, the results align with the literature for the PRED strategy for both types of items. However, we can see that groups of type DU (Dissimilar) present better results for the SG items when using the PROF aggregation strategy. These results show that when using different aggregation strategies and different types of items, there is a variation in the outcomes for the different group types.



(a) TOOLS (b) FOOD (c) MOVIE

Fig. 17. Metric results for different group types using NCF method and PROF aggregation strategy.

5.3 Statistical comparison

As we analyzed before, the major factors in experimental setting include aggregation strategies, aggregation functions, item types, and formation of groups (group types and sizes). This section utilizes statistical tests to analyze whether the choices of (a) aggregation strategies and (b) types of items make a difference in making recommendations. For both analysis, we use different group sizes (when fixing the group type) and group types (when fixing the group size). As in previous sections, we use the AVG aggregation function for all tests; SVD and NCF methods are chosen for the PRED and PROF aggregation strategies. The *t-test* analysis [93] is used.

5.3.1 Choice of aggregation strategies. The first set of statistical tests compares the effect of both the PRED and PROF aggregation strategies by testing all *group sizes* while fixing the group type to be RU. For each strategy (i.e., PRED and PROF) and each metric (i.e., HR, nDCG, Diversity, Coverage), 400 results are collected from experiments on 400 user groups (100 groups for each group size, four group sizes S, M, L, VL). Table 6 shows the results. We use as the null hypothesis, H_0 = *"the metric values of the recommendations generated by the PRED aggregation strategy equal the metric values of the recommendations generated by the PROF aggregation strategy for the group sizes"*. We evaluate if there is any significant difference with different aggregation strategies (when different group sizes are used) for TOOLS and FOOD datasets. We can see that there is no difference only for HR in TOOLS ($p > 0.05$) between both aggregation strategies. These results indicate that we should expect similar HR results using any aggregation strategy.

Table 6. Statistical t-test comparing the metrics obtained for the two aggregation strategies (PRED, PROF) for group sizes using TOOLS and FOOD datasets.

	TOOLS (PRED, PROF)		FOOD (PRED, PROF)	
	t-stat	p-value	t-stat	p-value
HR	-1.816	0.071 >0.05	-2.318	0.021 <0.05
nDCG	20.777	0.00 <0.05	15.762	0.00 <0.05
Diversity	-13.558	0.00 <0.05	-2.76	0.006 <0.05
Coverage	-4.101	0.00 <0.05	-3.682	0.00 <0.05

The second set of statistical tests compares the PRED and PROF aggregation strategies' effect by testing all *group types* while fixing group size to be S. Similar to the previous tests, we collect 400 results using the group types (i.e., RU, SU, DU, ReU). Table 7 shows the results. We use as the null hypothesis, H_0 = *"the metric values of the recommendations generated by the PRED aggregation strategy equal to the metric values of the recommendations generated by the PROF aggregation strategy for the group types"*.

We evaluate each dataset to see if there is any significant difference when different aggregation strategy is used for the group types. Only for HR in both datasets, there is no difference ($p > 0.05$) between both aggregation strategies.

Table 7. Statistical t-test comparing the metrics obtained for the two aggregation strategies (PRED, PROF) for group types for TOOLS and FOOD datasets.

	TOOLS (PRED, PROF)		FOOD (PRED, PROF)	
	t-stat	p-value	t-stat	p-value
HR	-0.119	0.906 >0.05	-1.271	0.205 >0.05
nDCG	16.783	0.00 <0.05	19.262	0.00 <0.05
Diversity	-12.268	0.00 <0.05	3.12	0.002 <0.05
Coverage	-2.619	0.009 <0.05	-3.173	0.002 <0.05

5.3.2 Choice of item types. This group of tests whether recommendation results are different where different types of items are used. We use the TOOLS dataset and the benchmark MOVIE dataset. This is mainly because the TOOLS dataset, a representative of the Amazon datasets, contains SG items and is less often used, while the MOVIE dataset is about EG items, is the preferred benchmark dataset in most *RS*.

The first test compares the effect of using datasets with different types of items by testing all *group sizes* while fixing the group type to be RU on each aggregation strategy (i.e., PRED and PROF). For each dataset and each metric, 400 results are collected from experiments on 400 user groups (100 groups for a given group size (i.e., S, M, L, VL)). Table 8 shows the results of the statistical t-tests. We use as the null hypothesis, H_0 = *"the metric values of the recommendations generated by the PRED (PROF) aggregation strategy for SG items equal the metric values of the recommendations generated by the PRED (PROF) aggregation strategy for EG items for the group sizes"*. We again observe that, only for Coverage for PROF aggregation strategy, there is no difference ($p > 0.05$) between both datasets.

Table 8. Statistical t-test comparing the metrics obtained on different item types (SG, EG) for the two aggregation strategies (PRED, PROF) for group sizes.

	PRED (TOOLS, MOVIE)		PROF (TOOLS, MOVIE)	
	t-stat	p-value	t-stat	p-value
HR	-12.512	0.00 <0.05	4.572	0.00 <0.05
nDCG	9.313	0.00 <0.05	5.831	0.00 <0.05
Diversity	62.745	0.00 <0.05	43.903	0.00 <0.05
Coverage	-12.628	0.00 <0.05	0.171	0.856 >0.05

The second test compares the effect of different types of datasets by testing all *group types* while fixing the group size to be S on each aggregation strategy. Like in the previous tests, we collected 400 results using the four group types (RU, SU, DU, ReU). Table 9 shows the results of the statistical t-tests. We use as the null hypothesis, H_0 = *"the metric values of the recommendations generated by the PRED (PROF) aggregation strategy for SG items equal the metric values of the recommendations generated by the PRED (PROF) aggregation strategy for EG items for the group types"*. We can see that there is no difference in Coverage ($p > 0.05$) between both datasets using the PROF aggregation strategy.

Table 9. Statistical t-test comparing the metrics obtained on different item types (SG, EG) for the two aggregation strategies (PRED, PROF) for group types.

	PRED (TOOLS, MOVIE)		PROF (TOOLS, MOVIE)	
	t-stat	p-value	t-stat	p-value
HR	-11.689	0.00 <0.05	4.218	0.00 <0.05
nDCG	11.256	0.00 <0.05	4.551	0.00 <0.05
Diversity	67.24	0.00 <0.05	47.882	0.00 <0.05
Coverage	-10.585	0.00 <0.05	0.03	0.976 >0.05

The results presented in this section show that there are evident differences between *GRecSys* using different aggregation strategies. These results indicate that performing the aggregation earlier or later in the recommendation process influences the final results. Besides, these results also indicate that there is a difference between the results using different types of items.

Based on these results, we can conclude that the recommendation methods show different behaviors on real-life datasets and on the MOVIE benchmark dataset.

6 CONCLUSIONS

We conducted detailed experimental comparisons of *GRecSys* by considering different factors, including the types of items to be recommended (SE or EG), the formation of groups (sizes and types), and the aggregation strategies (PRED and PROF) and functions. We found that 99.4% of the studies we assessed related to *GRecSys* evaluate their approach using EG items, where the MovieLens (MOVIE) dataset is the most used. Real-life datasets present different characteristics (i.e., sparsity, ratings per user, and ratings per item) from the widely used MOVIE dataset, and therefore the performance of *GRecSys* is affected.

For *GRecSys*, we found some valuable insights from our experiments. First, the type of items (i.e., SG and EG) influence the results obtained from *GRecSys*. As far as we know, this analysis had not been explored before. Second, when different types of groups are used, we found a clear difference

in the recommendation results when different aggregation strategies are used in these user groups. Third, recommendations become more difficult as the group grows. This finding is consistent with the literature. Fourth, SVD and NCF are the recommender system algorithms presenting the best performance for real-life datasets using PRED and PROF aggregation strategies, respectively. Fifth, the AVG aggregation function presents higher performance than the other aggregation functions, and it is aligned with previous works. Finally, using different aggregation strategies and different types of items make a statistical difference in recommendation results.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2011. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2011), 896–911.

[2] Akshita Agarwal, Manajit Chakraborty, and C Ravindranath Chowdary. 2017. Does order matter? Effect of order in group recommendation. *Expert Systems with Applications* 82 (2017), 115–127.

[3] Hyung Jun Ahn. 2008. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences* 178, 1 (2008), 37–51.

[4] Bushra Alhijawi, Ghazi Al-Naymat, Nadim Obeid, and Arafat Awajan. 2019. Mitigating the Effect of Data Sparsity: A Case Study on Collaborative Filtering Recommender System. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE, 1–6.

[5] Irfan Ali and Sang-Wook Kim. 2015. An effective approach to group recommendation based on belief propagation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 1148–1153.

[6] Frederick Ayala-Gómez, Bálint Zoltán Daróczy, Michael Mathioudakis, András Benczúr, and Aristides Gionis. 2017. Where could we go? Recommendations for groups in location-based social networks. (2017).

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

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

[9] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. 2010. Space efficiency in group recommendation. *The VLDB Journal—The International Journal on Very Large Data Bases* 19, 6 (2010), 877–900.

[10] Daniel J Beal, Robin R Cohen, Michael J Burke, and Christy L McLendon. 2003. Cohesion and performance in groups: a meta-analytic clarification of construct relations. *Journal of applied psychology* 88, 6 (2003), 989.

[11] Shlomo Berkovsky and Jill Freyne. 2010. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 111–118.

[12] Shlomo Berkovsky and Jill Freyne. 2010. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 111–118.

[13] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[14] Robin Burke. 2007. Hybrid web recommender systems. In *The adaptive web*. Springer, 377–408.

[15] Lucas Augusto Montalvão Costa Carvalho and Hendrik Teixeira Macedo. 2013. Users' satisfaction in recommendation systems for groups: an approach based on noncooperative games. In *Proceedings of the 22nd International Conference on World Wide Web*. 951–958.

[16] Jorge Castro, Jie Lu, Guangquan Zhang, Yucheng Dong, and Luis Martínez. 2017. Opinion dynamics-based group recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 12 (2017), 2394–2406.

[17] Jorge Castro, Jie Lu, Guangquan Zhang, Yucheng Dong, and Luis Martínez. 2017. Opinion dynamics-based group recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 12 (2017), 2394–2406.

[18] Edgar Ceh-Varela and Huiping Cao. 2019. Recommending Packages of Multi-Criteria Items to Groups. In *2019 IEEE International Conference on Web Services (ICWS)*. IEEE, 273–282.

[19] Robert B Cialdini and Noah J Goldstein. 2004. Social influence: Compliance and conformity. *Annu. Rev. Psychol.* 55 (2004), 591–621.

[20] Toon De Pessemier, Jeroen Dhondt, Kris Vanhecke, and Luc Martens. 2015. TravelWithFriends: a hybrid group recommender system for travel destinations. In *Workshop on tourism recommender systems (touRS15), in conjunction with the 9th ACM conference on recommender systems (recsys 2015)*. 51–60.

[21] Toon De Pessemier, Simon Dooms, and Luc Martens. 2012. Design and evaluation of a group recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 225–228.

[22] Toon De Pessemier, Simon Dooms, and Luc Martens. 2013. An improved data aggregation strategy for group recommendations. In *3rd Workshop on Human Decision Making in Recommender Systems (Decisions@ RecSys 2013), held in conjunction with the 7th ACM Conference on Recommender Systems (RecSys 2013)*, Vol. 1050. 36–39.

[23] Amra Delic, Francesco Ricci, and Julia Neidhardt. 2019. Preference Networks and Non-Linear Preferences in Group Recommendations. In *IEEE/WIC/ACM International Conference on Web Intelligence*. ACM, 403–407.

[24] Danhao Ding, Hui Li, Zhipeng Huang, and Nikos Mamoulis. 2017. Efficient fault-tolerant group recommendation using alpha-beta-core. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2047–2050.

[25] Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič. 2018. Evaluating group recommender systems. In *Group Recommender Systems*. Springer, 59–71.

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

[27] Guillermo Fernández, Waldemar López, Fernando Olivera, Bruno Rienzi, and Pablo Rodríguez-Bocca. 2014. Let's go to the cinema! A movie recommender system for ephemeral groups of users. In *2014 XL Latin American Computing Conference (CLEI)*. IEEE, 1–12.

[28] Noah E Friedkin. 2004. Social cohesion. *Annu. Rev. Sociol.* 30 (2004), 409–425.

[29] Baode Gao, Guangpeng Zhan, Hanzhang Wang, Yiming Wang, and Shengxin Zhu. 2019. Learning with Linear Mixed Model for Group Recommendation Systems. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*. ACM, 81–85.

[30] Inma Garcia, Laura Sebastia, and Eva Onaindia. 2011. On the design of individual and group recommender systems for tourism. *Expert systems with applications* 38, 6 (2011), 7683–7692.

[31] Mike Gartrell, Xinyu Xing, Qin Lv, Aaron Beach, Richard Han, Shivakant Mishra, and Karim Seada. 2010. Enhancing group recommendation by incorporating social relationship interactions. In *Proceedings of the 16th ACM international conference on Supporting group work*. 97–106.

[32] Sarik Ghazarian and Mohammad Ali Nematbakhsh. 2015. Enhancing memory-based collaborative filtering for group recommender systems. *Expert systems with applications* 42, 7 (2015), 3801–3812.

[33] SongJie Gong, HongWu Ye, and HengSong Tan. 2009. Combining memory-based and model-based collaborative filtering in recommender system. In *2009 Pacific-Asia Conference on Circuits, Communications and Systems*. IEEE, 690–693.

[34] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[35] Miao He, Weixi Gu, and Ying Kong. 2017. Group recommendation: by mining users' check-in behaviors. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM, 65–68.

[36] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

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

[38] Daniel Herzog and Wolfgang Wörndl. 2019. User-centered evaluation of strategies for recommending sequences of points of interest to groups. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, 96–100.

[39] Daniel Herzog and Wolfgang Wörndl. 2019. A User Study on Groups Interacting with Tourist Trip Recommender Systems in Public Spaces. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*. ACM, 130–138.

[40] Xun Hu, Xiangwu Meng, and Licai Wang. 2011. Svd-based group recommendation approaches: an experimental study of moviepilot. In *Proceedings of the 2nd challenge on context-aware movie recommendation*. 23–28.

[41] Peng Huang, Nicholas H Lurie, and Sabyasachi Mitra. 2009. Searching for experience on the web: an empirical examination of consumer behavior for search and experience goods. *Journal of marketing* 73, 2 (2009), 55–69.

[42] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[43] Won-Seok Hwang, Juan Parc, Sang-Wook Kim, Jongwuk Lee, and Dongwon Lee. 2016. "Told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 349–360.

[44] Hyun Ji Jeong and Myoung Ho Kim. 2019. HGGC: A hybrid group recommendation model considering group cohesion. *Expert Systems with Applications* 136 (2019), 73–82.

[45] Venkateswara Rao Kagita, Krishna Charan Meka, and Vineet Padmanabhan. 2015. A novel social-choice strategy for group modeling in recommender systems. In *2015 International Conference on Information Technology (ICIT)*. IEEE, 153–158.

[46] Ondrej Kaššák, Michal Kompan, and Mária Bieliková. 2016. Personalized hybrid recommendation for group of users: Top-N multimedia recommender. *Information Processing & Management* 52, 3 (2016), 459–477.

[47] Heung-Nam Kim and Abdulmotaleb El Saddik. 2015. A stochastic approach to group recommendations in social media systems. *Information Systems* 50 (2015), 76–93.

[48] Bart P Knijnenburg, Martijn C Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* 22, 4-5 (2012), 441–504.

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

[50] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems–A survey. *Knowledge-Based Systems* 123 (2017), 154–162.

[51] Ricardo Lage, Frederico Durao, and Peter Dolog. 2012. Towards effective group recommendations for microblogging users. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. 923–928.

[52] Arthur S Leahy. 2005. Search and Experience Goods: Evidence from the 1960's and 70's. *Journal of Applied Business Research (JABR)* 21, 1 (2005).

[53] Daniel Lemire and Anna Maclachlan. 2005. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 471–475.

[54] Guohui Li, Qi Chen, Bolong Zheng, Hongzhi Yin, Quoc Viet Hung Nguyen, and Xiaofang Zhou. 2020. Group-Based Recurrent Neural Networks for POI Recommendation. *ACM Transactions on Data Science* 1, 1 (2020), 1–18.

[55] Ruichang Li, Honglei Zhu, Liao Fan, and Xuekun Song. 2019. Hybrid Deep Framework for Group Event Recommendation. *IEEE Access* 8 (2019), 4775–4784.

[56] Tao Liu, Feng Xu, Yuan Yao, and Jian Lu. 2012. A group recommendation approach for service selection. In *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*. 1–5.

[57] Ziyu Lu, Hui Li, Nikos Mamoulis, and David W Cheung. 2017. Hbgg: a hierarchical Bayesian geographical model for group recommendation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 372–380.

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

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

[60] Matthew R McLaughlin and Jonathan L Herlocker. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 329–336.

[61] Hanan Mengash and Alexander Brodsky. 2016. Tailoring Group Package Recommendations to Large Heterogeneous Groups Based on Multi-Criteria Optimization. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 1537–1546.

[62] Susan M Mudambi and David Schuff. 2010. What makes a helpful review? A study of customer reviews on Amazon. com. *MIS quarterly* 34, 1 (2010), 185–200.

[63] Lihi Naamani Dery, Meir Kalech, Lior Rokach, and Bracha Shapira. 2010. Iterative voting under uncertainty for group recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 265–268.

[64] Lihi Naamani-Dery, Meir Kalech, Lior Rokach, and Bracha Shapira. 2014. Preference elicitation for narrowing the recommended list for groups. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, 333–336.

[65] Reza Barzegar Nozari and Hamidreza Koohi. 2020. A novel group recommender system based on members' influence and leader impact. *Knowledge-Based Systems* 205 (2020), 106296.

[66] Fernando Ortega, Antonio Hernando, Jesus Bobadilla, and Jeon Hyung Kang. 2016. Recommending items to group of users using matrix factorization based collaborative filtering. *Information Sciences* 345 (2016), 313–324.

[67] Fernando Ortega, Remigio Hurtado, Jesus Bobadilla, and Rodolfo Bojorque. 2018. Recommendation to groups of users using the singularities concept. *IEEE Access* 6 (2018), 39745–39761.

[68] Zacharoula Papamitsiou and Anastasios A Economides. 2018. Can't get more satisfaction?: game-theoretic group-recommendation of educational resources. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. ACM, 409–416.

[69] Shameem A Puthiya Parambath, Nishant Vijayakumar, and Sanjay Chawla. 2018. Saga: A submodular greedy algorithm for group recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[70] Shameem A Puthiya Parambath, Nishant Vijayakumar, and Sanjay Chawla. 2018. Saga: A submodular greedy algorithm for group recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[71] Alexander Pelaez, Martin Y Yu, and Karl R Lang. 2013. Social buying: the effects of group size and communication on buyer performance. *International journal of electronic commerce* 18, 2 (2013), 127–157.

[72] Maria S Pera and Yiu-Kai Ng. 2013. A group recommender for movies based on content similarity and popularity. *Information Processing & Management* 49, 3 (2013), 673–687.

[73] Abinash Pujahari and Vineet Padmanabhan. 2015. Group Recommender Systems: Combining user-user and item-item Collaborative filtering techniques. In *2015 International Conference on Information Technology (ICIT)*. IEEE, 148–152.

[74] Abinash Pujahari and Dilip Singh Sisodia. 2020. Aggregation of Preference Relations to Enhance the Ranking Quality of Collaborative Filtering based Group Recommender System. *Expert Systems with Applications* (2020), 113476.
[75] Sanjay Purushotham and C-C Jay Kuo. 2016. Personalized group recommender systems for location-and event-based social networks. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 2, 4 (2016), 16.
[76] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Recommending packages to groups. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 449–458.
[77] Dong Qin, Xiangmin Zhou, Lei Chen, Guangyan Huang, and Yanchun Zhang. 2018. Dynamic connection-based social group recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).
[78] Lara Quijano-Sanchez, Juan A Recio-Garcia, Belen Diaz-Agudo, and Guillermo Jimenez-Diaz. 2013. Social factors in group recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 4, 1 (2013), 1–30.
[79] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. 2016. Recommending new items to ephemeral groups using contextual user influence. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 285–292.
[80] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. 2016. Recommending new items to ephemeral groups using contextual user influence. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 285–292.
[81] Dimitris Sacharidis. 2017. Group recommendations by learning rating behavior. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 174–182.
[82] Dimitris Sacharidis. 2019. Modeling Uncertainty in Group Recommendations. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*. 69–74.
[83] Dimitris Sacharidis. 2019. Top-N group recommendations with fairness. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 1663–1670.
[84] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. 791–798.
[85] Maria Salamó, Kevin McCarthy, and Barry Smyth. 2012. Generating recommendations for consensus negotiation in group personalization services. *Personal and Ubiquitous Computing* 16, 5 (2012), 597–610.
[86] Amirali Salehi-Abari and Craig Boutilier. 2015. Preference-oriented social networks: Group recommendation and inference. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 35–42.
[87] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1 (2001), 285–295.
[88] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.
[89] Young-Duk Seo, Young-Gab Kim, Euijong Lee, Kwang-Soo Seol, and Doo-Kwon Baik. 2018. An enhanced aggregation method considering deviations for a group recommendation. *Expert Systems with Applications* 93 (2018), 299–312.
[90] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2017. Fairness in package-to-group recommendations. In *Proceedings of the 26th International Conference on World Wide Web*. 371–379.
[91] Jing Shi, Bin Wu, and Xiuqin Lin. 2015. A latent group model for group recommendation. In *2015 IEEE International conference on mobile services*. IEEE, 233–238.
[92] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. 2019. How good your recommender system is? A survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics* 10, 5 (2019), 813–831.
[93] Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. 623–632.
[94] Maria Stratigi, Jyrki Nummenmaa, Evaggelia Pitoura, and Kostas Stefanidis. 2020. Fair sequential group recommendations. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 1443–1452.
[95] Florian Strub and Jeremie Mary. 2015. Collaborative filtering with stacked denoising autoencoders and sparse inputs.
[96] Lifeng Sun, Xiaoyan Wang, Zhi Wang, Hong Zhao, and Wenwu Zhu. 2016. Social-aware video recommendation for online social groups. *IEEE Transactions on Multimedia* 19, 3 (2016), 609–618.
[97] Poonam B Thorat, RM Goudar, and Sunita Barve. 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications* 110, 4 (2015), 31–36.
[98] Joseph K Torgesen. 2006. Intensive Reading Interventions for Struggling Readers in Early Elementary School: A Principal's Guide. *Center on Instruction* (2006).
[99] Oren Tsur and Ari Rappoport. 2009. Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews. In *Third International AAAI Conference on Weblogs and Social Media*.
[100] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 501–508.

[101] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 515–524.

[102] Ximeng Wang, Yun Liu, Jie Lu, Fei Xiong, and Guangquan Zhang. 2019. TruGRC: Trust-Aware Group Recommendation with Virtual Coordinators. *Future Generation Computer Systems* 94 (2019), 224–236.

[103] Ximeng Wang, Yun Liu, Jie Lu, Fei Xiong, and Guangquan Zhang. 2019. TruGRC: Trust-Aware Group Recommendation with Virtual Coordinators. *Future Generation Computer Systems* 94 (2019), 224–236.

[104] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, Vol. 8. 6.

[105] Caihua Wu, Junwei Wang, Juntao Liu, and Wenyu Liu. 2016. Recurrent neural network based recommendation for time heterogeneous feedback. *Knowledge-Based Systems* 109 (2016), 90–103.

[106] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. Npa: Neural news recommendation with personalized attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2576–2584.

[107] Zhengzheng Xian, Qiliang Li, Gai Li, and Lei Li. 2017. New Collaborative Filtering Algorithms Based on SVD. *Mathematical Problems in Engineering* 2017 (2017).

[108] Hongzhi Yin, Qinyong Wang, Kai Zheng, Zhixu Li, and Xiaofang Zhou. 2020. Overcoming Data Sparsity in Group Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[109] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: a generative model for group recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 163–172.

[110] Ting Yuan, Jian Cheng, Xi Zhang, Shuang Qiu, and Hanqing Lu. 2014. Recommendation by mining multiple user behaviors with group sparsity. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[111] Alfredo Zapata, Víctor H Menéndez, Manuel E Prieto, and Cristóbal Romero. 2015. Evaluation and selection of group recommendation strategies for collaborative searching of learning objects. *International Journal of Human-Computer Studies* 76 (2015), 22–39.

[112] Chen Zhang, Jing Zhou, and Weifeng Xie. 2016. A users clustering algorithm for group recommendation. In *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)*. IEEE, 352–356.

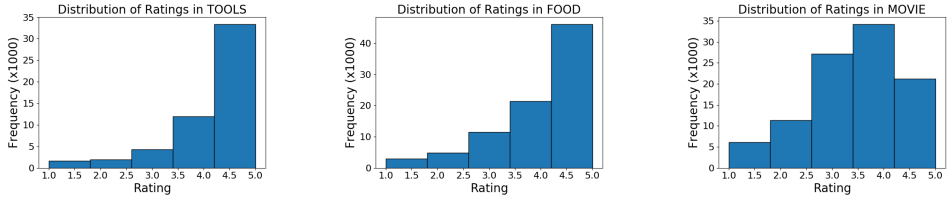
[113] Jason Shuo Zhang, Mike Gartrell, Richard Han, Qin Lv, and Shivakant Mishra. 2019. GEVR: An Event Venue Recommendation System for Groups of Mobile Users. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 34.

[114] Yong Zheng. 2020. Educational Group Recommendations with Virtual Leaders. In *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 52–56.

[115] Haiping Zhu, Yifu Ni, Feng Tian, Pei Feng, Yan Chen, and Qinghua Zheng. 2018. A group-oriented recommendation algorithm based on similarities of personal learning generative networks. *IEEE Access* 6 (2018), 42729–42739.

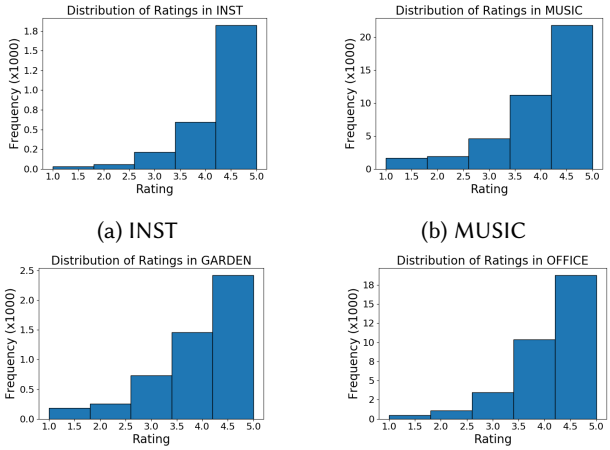
[116] Qiliang Zhu and Lei Wang. 2020. Context-Aware Restaurant Recommendation for Group of People. In *2020 IEEE World Congress on Services (SERVICES)*. IEEE, 51–54.

APPENDIX



(a) TOOLS (b) FOOD (c) MOVIE

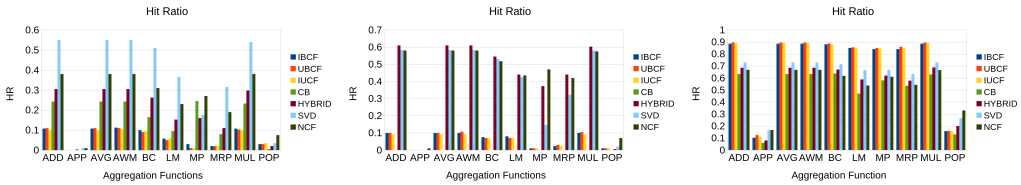
Fig. 18. Dataset ratings distribution.



(a) INST (b) MUSIC

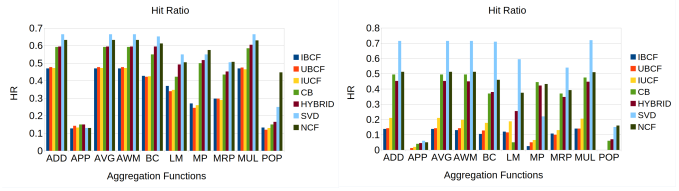
(c) GARDEN (d) OFFICE

Fig. 19. Dataset ratings distribution.

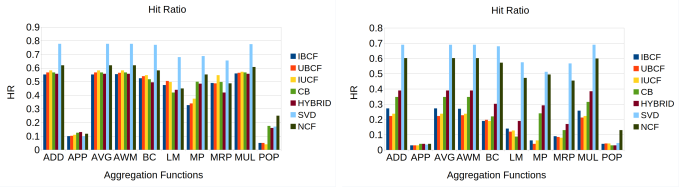


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 20. HR metric for all recommender systems using PRED aggregation strategy



(a) INST (b) MUSIC

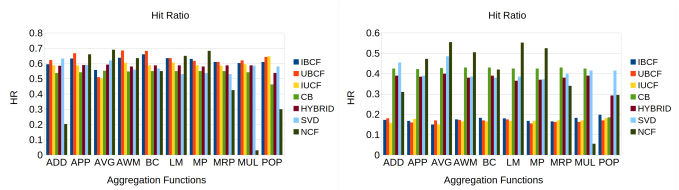


(c) GARDEN (d) OFFICE

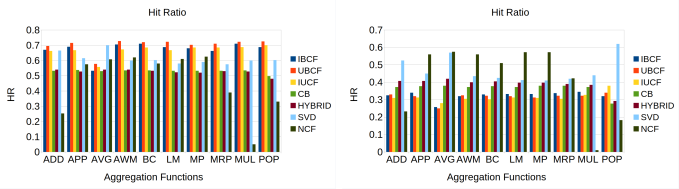
Fig. 21. HR metric for all recommender systems using PRED aggregation strategy

(a) TOOLS (b) FOOD (c) MOVIE

Fig. 22. HR metric for all recommender systems using PROF aggregation strategy

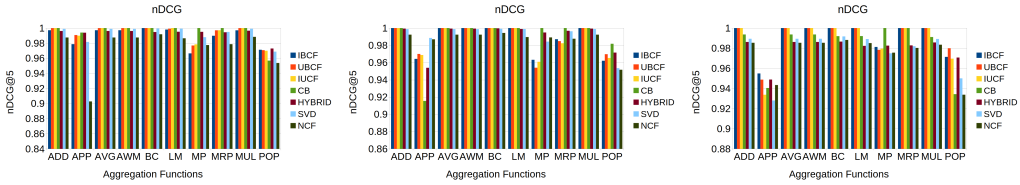


(a) INST (b) MUSIC



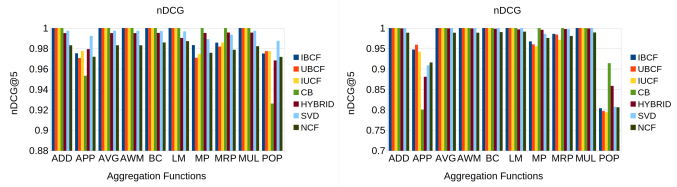
(c) GARDEN (d) OFFICE

Fig. 23. HR metric for all recommender systems using PROF aggregation strategy

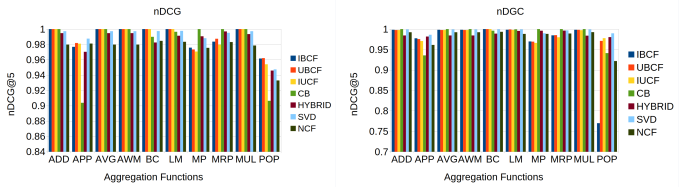


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 24. nDCG@5 metric for all recommender systems using PRED aggregation strategy

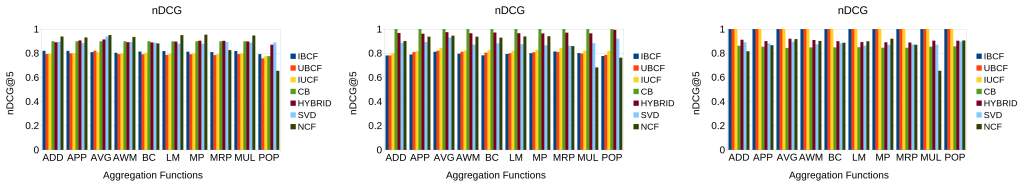


(a) INST (b) MUSIC



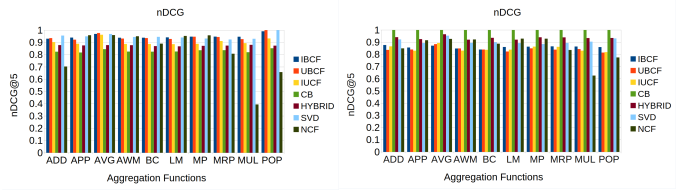
(c) GARDEN (d) OFFICE

Fig. 25. nDCG@5 metric for all recommender systems using PRED aggregation strategy

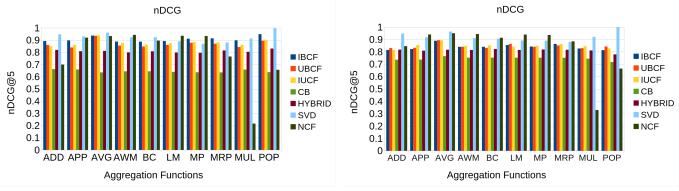


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 26. nDCG@5 metric for all recommender systems using PROF aggregation strategy

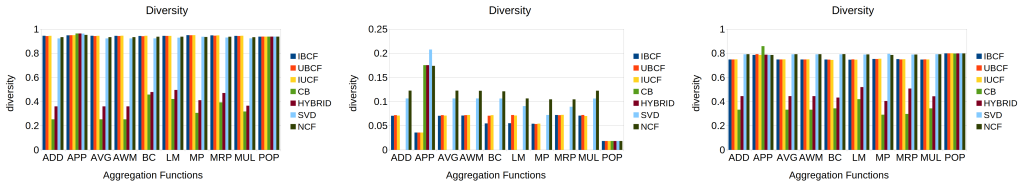


(a) INST (b) MUSIC



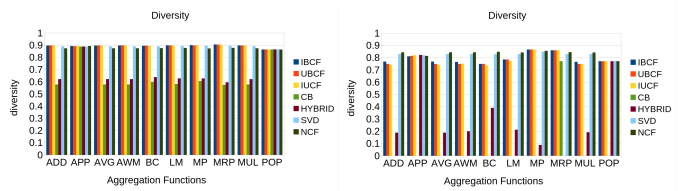
(c) GARDEN (d) OFFICE

Fig. 27. nDCG@5 metric for all recommender systems using PROF aggregation strategy

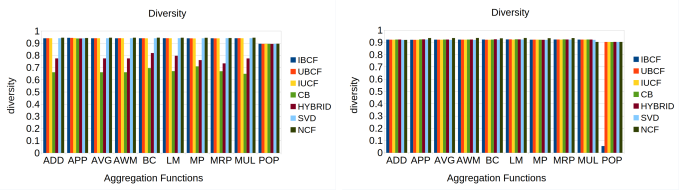


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 28. Diversity metric for all recommender systems using PRED aggregation strategy

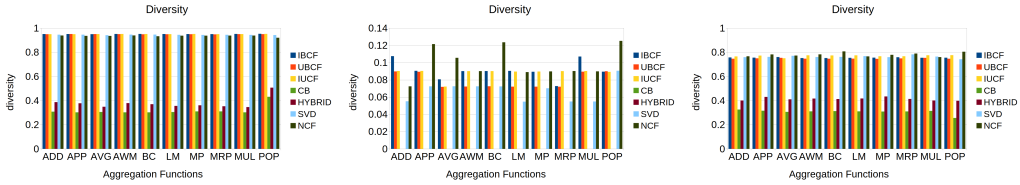


(a) INST (b) MUSIC



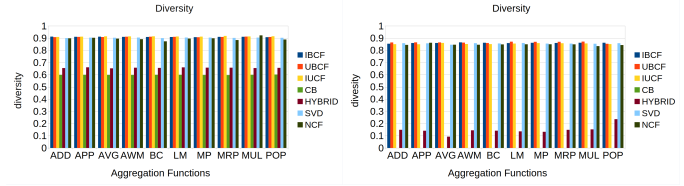
(c) GARDEN (d) OFFICE

Fig. 29. Diversity metric for all recommender systems using PRED aggregation strategy

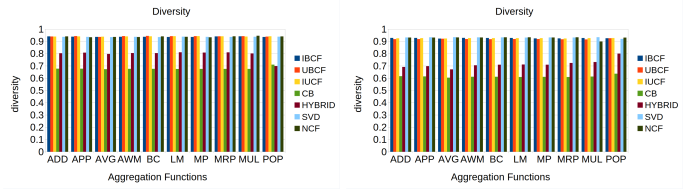


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 30. Diversity metric for all recommender systems using PROF aggregation strategy

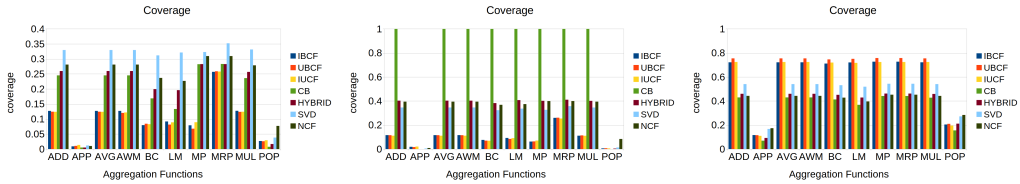


(a) INST (b) MUSIC



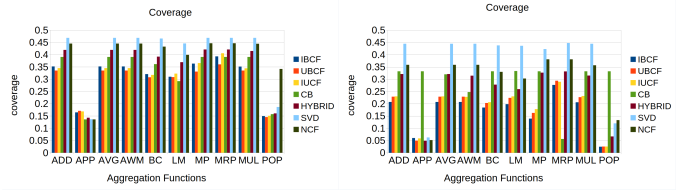
(c) GARDEN (d) OFFICE

Fig. 31. Diversity metric for all recommender systems using PROF aggregation strategy

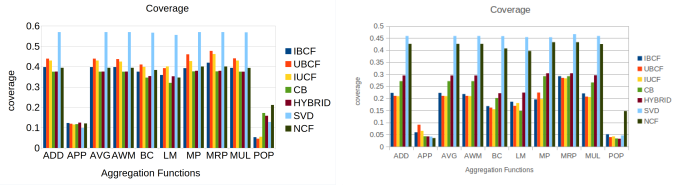


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 32. Coverage metric for all recommender systems using PRED aggregation strategy

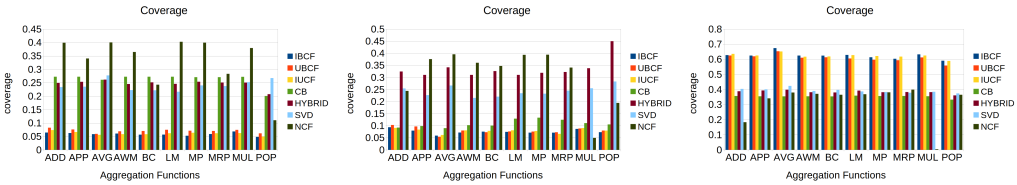


(a) INST (b) MUSIC



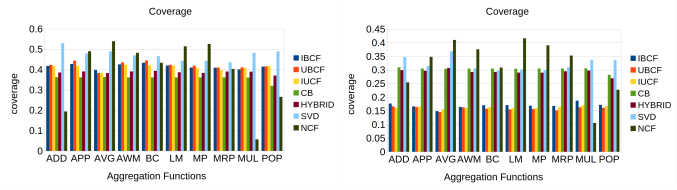
(c) GARDEN (d) OFFICE

Fig. 33. Coverage metric for all recommender systems using PRED aggregation strategy

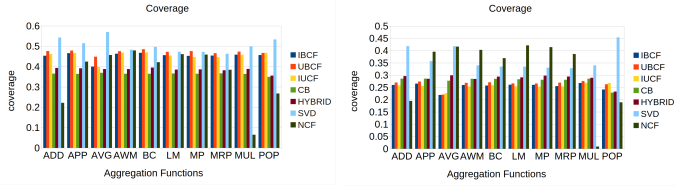


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 34. Coverage metric for all recommender systems using PROF aggregation strategy

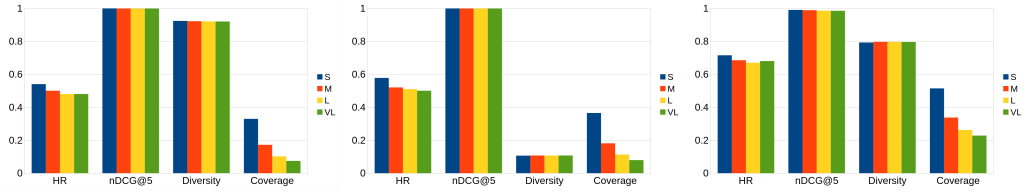


(a) INST (b) MUSIC



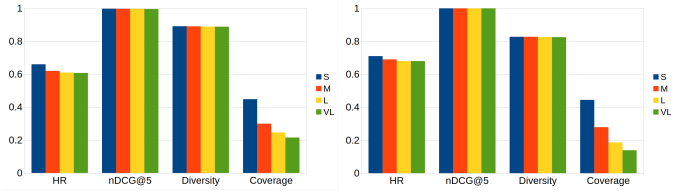
(c) GARDEN (d) OFFICE

Fig. 35. Coverage metric for all recommender systems using PROF aggregation strategy

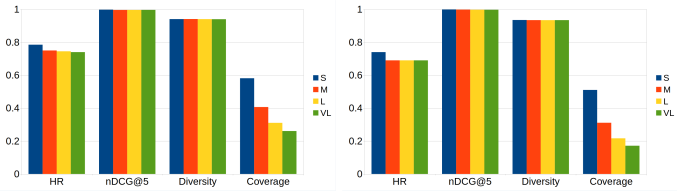


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 36. Metric results for different group sizes using SVD method and PRED aggregation strategy

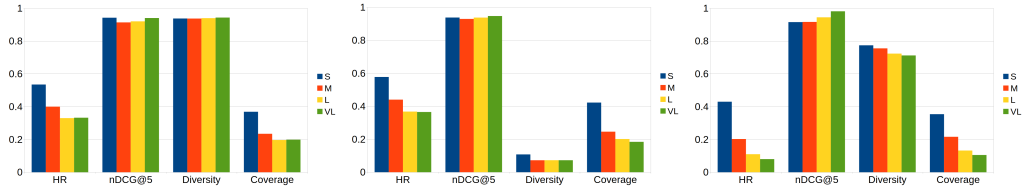


(a) INST (b) MUSIC



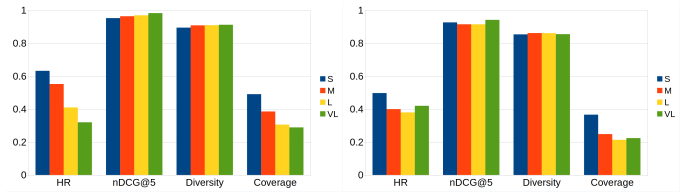
(c) GARDEN (d) OFFICE

Fig. 37. Metric results for different group sizes using SVD method and PRED aggregation strategy

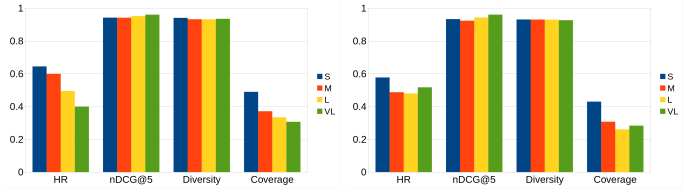


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 38. Metric results for different group sizes using NCF method and PROF aggregation strategy

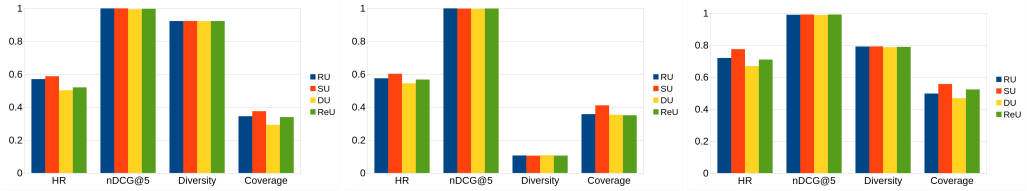


(a) INST (b) MUSIC



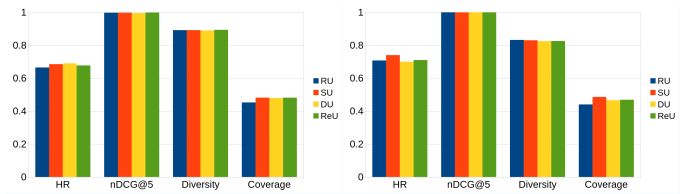
(c) GARDEN (d) OFFICE

Fig. 39. Metric results for different group sizes using NCF method and PROF aggregation strategy

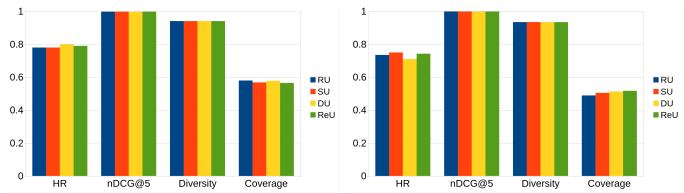


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 40. Metric results for different group types using SVD method and PRED aggregation strategy

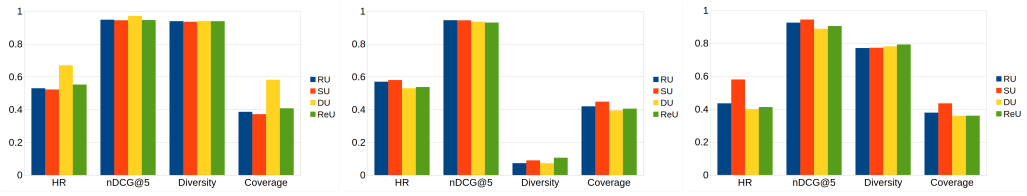


(a) INST (b) MUSIC



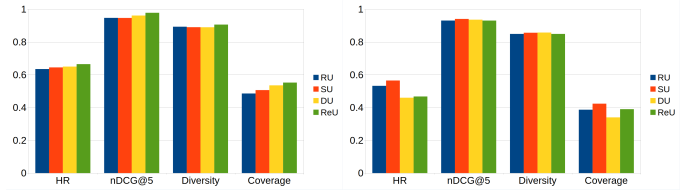
(c) GARDEN (d) OFFICE

Fig. 41. Metric results for different group types using SVD method and PRED aggregation strategy

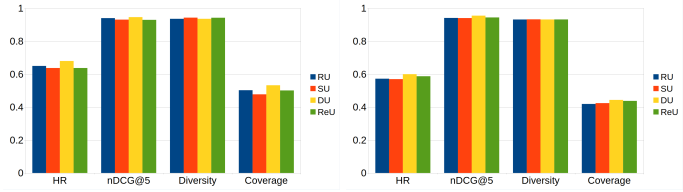


(a) TOOLS (b) FOOD (c) MOVIE

Fig. 42. Metric results for different group types using NCF method and PROF aggregation strategy



(a) INST (b) MUSIC



(c) GARDEN (d) OFFICE

Fig. 43. Metric results for different group types using NCF method and PROF aggregation strategy