Evaluation of Algorithms for Group Recommendation Systems with Top-K Restrictions.

Leonardo Antonetti da Motta [a], Ladislav Peska [b].

[a] Institute of Mathematical And Computer Sciences, University of São Paulo, São Carlos, Brasil, l.a.motta@usp.br

[b] Faculty of Mathematics and Physics, Charles University, Malostranské nám. 25, Prague, Czech Republic, ladislav.peska@matfyz.cuni.cz

Abstract. Recommender Systems (RS) are used as a means to assist users in getting proper recommendations, when considering various items that can be matched to an individual user. Group Recommender Systems (GRS) generally work off of the results of RS, under the lenses of a group of users requiring a unified list of recommendations that satisfies all those users to a certain extent. The performance of GRS has been relatively varied across different group recommendation algorithms, and between coupled and decoupled modes of evaluation. This is especially evident when GRS suffers from lack of explicit information regarding individual users within a group. We use a previously implemented set of GRS algorithms while also implementing three additional algorithms to this set, and evaluate the whole set of algorithms with multiple randomly chosen groups with varying individual user ratings. We repeat the results with different top-k restrictions on individual user ratings to observe the impact of limited user information on each algorithm. While lower top-k values favoured algorithms such as MPL and EPFuzzDA, higher top-k values favoured algorithms such as MUL and LMS. When using top-k restrictions that were below the total number of expected recommendations, the MUL, LMS, and GFAR algorithms could not offer significant results. Algorithms such as MPL, FAI and AVGNM remained relatively consistent regardless of top-k restrictions. These results, when compiled, could be skewed by different definitions of 'fairness' in regards to evaluation. Regardless, the top-k restriction has a significant impact on the performance of a given group recommendation algorithm, considering the metrics observed.

Keywords. Group Recommendations, Recommender Systems, Fairness, Evaluation.

1. Introduction

In recent years, Recommender Systems (RS) have gained significant attention as a means to assist users in finding relevant and personalized items, such as movies, books, products, and services. While traditional RS focus on generating recommendations for individual users, Group Recommender Systems (GRS) aim to generate recommendations for a group of users, taking into account their collective preferences and needs. According to Konstan and Riedl [1], GRS can be used in various contexts, such as social networks, e-commerce platforms, and collaborative work environments, where recommendations that satisfy the preferences and interests of all group members are crucial for enhancing user satisfaction and engagement. However, designing effective and fair group recommender systems poses several challenges, as pointed out by Shani and Gunawardana [2], such as the need to balance between individual and group preferences, handle potential conflicts and diversity among group members, and ensure fairness and transparency in the recommendation process.

There have been many attempts to delineate algorithms for GRS and define proper methods of evaluating fairness in the group recommendations. Masthoff [3] offers a possible definition of fairness by observing how participants attempted to avoid misery and starvation (in regards to individual user preference), which reflected some algorithms such as 'Average Least Misery' and 'Average Without Misery'. These values of utility would then be assigned on a per-item basis.

This principle was also somewhat reflected in a

recently proposed method by Sacharidis [4], who defines the most fair list as a list that minimizes the dissatisfaction of any observed group member, enforcing the Least Misery principle among group utility. Group utility being defined as the average member utility, and fairness as the minimum member utility. However, it should be noted that for Sacharidis' fairness methodology, the values of utility are assigned on a per-list basis.

However, it is important to note the caveat delineated by Kaya [5], where it's noted that current methods do not define fairness in a rank-sensitive way. Kaya pointed out that, although Least Misery might replicate the most common human use of the idea of fairness, it does not consider if the relevance of the top-k items is balanced for users for each prefix of the top-k, for example. In other words, if you observe the first item alone, it should balance the interests of all users. Then, for the top two items taken together, the same must occur, and so on for all top-k selections.

Other papers' findings [6, 7] also make a strong case for rank-sensitive definitions of fairness, considering how attention towards top-displayed items tends to be higher, and lower for last-displayed items. A rank-sensitive approach would equalize this uneven distribution of attention, while list-wise and item-wise approaches might not reach similar results.

One of the current challenges of GRS is performance over large sets of data, and incomplete user-item pairs. It is common to have those limitations in a RS pipeline, and therefore it's expected that similar restrictions would be present in GRS (i.e. lacking the budget to evaluate preferences of all user-item pairs, or lacking information about all user-item pairs).

Furthermore, the various methods of evaluating results—such as coupled and decoupled evaluation as delineated by Peska and Malecek [8]—can provide highly diverse findings considering both approaches have underlying varying ratios of known *vs.* unknown information, making it difficult to choose the most appropriate method for each scenario.

In this paper, we implement and review the more prominent algorithms in GRS, test different top-k limitations for the available data sent to the GRS, check the final results with varying metrics in both coupled and decoupled evaluations, and observe possible challenges of each approach.

2. Methodology

The pipeline of data to results and evaluation is demonstrated in Fig. 1. In summary, user-item data was collected, groups of users were generated from that data, recommendations for each group were created based on nine different algorithms, then finally those recommendations were evaluated based on six different metrics, and this evaluation was repeated for both coupled and decoupled scenarios. More details are explained in the following sections.

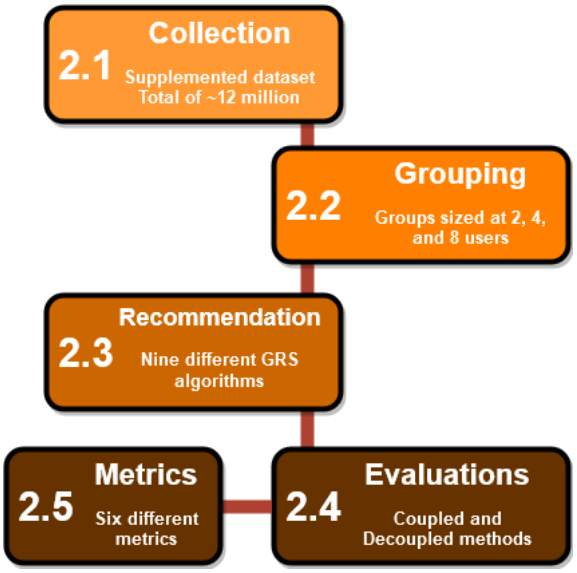

Fig. 1 - Pipeline delineating the progression of all steps taken in this paper, from data towards final results. Numbers used are the same as the section numbers detailing the steps.

2.1 Data Collection

The basis of data was the MovieLens 1 million dataset (ML-1M), which has 1 million ratings from 6000 users on 4000 movies, with a rating range of 0 to 5 (higher equals better rating). However, the dataset was missing certain ratings for some user-movie pairings. Thus, the dataset was supplemented with predicted ratings based on previous explicit ratings for each user, for a total of approximately 12 million rows of ratings (6000 users with a total of 2020 ratings each). The predictions were made using stratified K-Fold division and prediction methods from an individual RS, that are also later used as ground-truth for decoupled evaluation.

With the full ratings file, various top-k restricted files were created. The process of top-k restriction iterates through the full ratings file, gets all per-user ratings and orders them from highest to lowest. Then it converts every rating except the top K ratings of each user to zero. This effectively means that the top-k restricted file only has true access to the K-highest ratings of each user, and all other ratings are zero. The top-k restricted files were created as indicated by Tab. 1.

Tab. 1 - Definition of divisions for each top-k file, and their true item total.

Name of file	K value	Percentage of total user ratings
'Topk-1'	1	0.05 %
'Topk-5'	5	0.25 %
'Topk-10'	10	0.5 %

'Topk-21'	21	1 %
'Topk-202'	202	10 %
'Topk-505'	505	50 %
'Topk-1010'	1010	75 %
'Topk-2020'	2020	100 %

2.2 Grouping

Groups were synthetically generated with different sizes (2, 4, and 8 users each group) and randomly selected users. The groupings were tested for similarity with Pearson's Correlation Coefficient, as can be seen in Fig. 2.

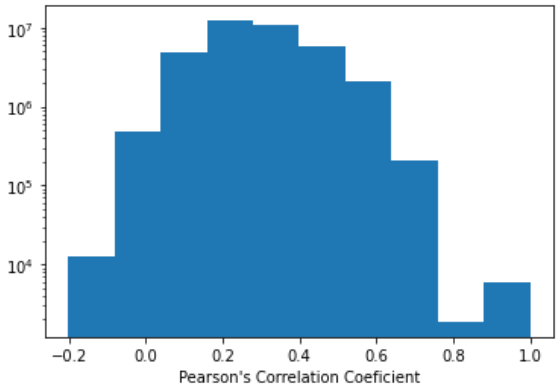

Fig. 2 - Correlation Coefficient of randomly generated groups, with average similarity values used for separation.

2.3 Recommendations

Recommendations were limited by a total of ten per group. Each group had ten item recommendations given for each of the nine chosen GRS aggregator algorithms.

Additive Algorithm (ADD) [3] and the **Multiplicative Algorithm (MUL)** [3] are similar in their nature. Both aggregate the item ratings, through addition and multiplication respectively, and return the top items after ordering the results from higher to lower ratings.

Least Misery Algorithm (LMS) [3] aggregates the items by taking the minimum rating found in the total ratings set from all users within the group, and returns the top items. It is a conservative approach to decision-making, as it aims to minimize the potential for negative outcomes rather than maximizing potential benefits. It is particularly useful in situations where the consequences of a decision could be significant and negative, such as in medical or safety-critical applications. However, it may not be appropriate in situations where it is important to maximize positive outcomes or where the consequences of a negative outcome are less severe.

Most Pleasure Algorithm (MPL) [3] aggregates the items by taking the maximum rating found in the total ratings set from all users within the group, and returns the top items. It is a more optimistic approach to decision-making than the Least Misery Algorithm, as it aims to maximize positive outcomes rather than minimize negative ones. It is particularly useful in situations where the consequences of a decision are primarily positive, such as in marketing or product design. However, it may not be appropriate in situations where negative outcomes could be severe or where it is important to balance multiple criteria with different levels of importance.

GFAR Algorithm (GFAR) [5] is determined by calculating the sum of probabilities that at least one recommended item is relevant for the user. This is represented as the complement of the probability that all items are irrelevant. Relevance probabilities of individual items are defined as the normalized Borda-count induced by the individual preferences of user u. GFAR follows a greedy approach to generate its list in an iterative manner. However, GFAR tends to select items that are per-user best rather than focusing on items that have an (certain level of) overall agreement, as it relies on a somewhat extreme estimation of relevance probability and assumes that a single relevant item per user is fair enough. [8].

EPFuzzDA Algorithm (EPFuzzDA) [9] is the approach that, for each user and each iteration, EP-FuzzDA determines the relevance of items in a list based on the proportional representation of user's votes. It calculates the amount of (constrained) relevance that is required to achieve proportional representation and then selects the item with the highest overall relevance, taking into account certain constraints. Unlike other methods, EP-FuzzDA tends to prioritize items with higher overall agreement rather than focusing on the best items for individual users. [8].

Fairness Algorithm (FAI) [3] simulates each individual user picking items in turn based on their own ordered ranking of preference, without consideration for any other factors. User order is randomly selected each time. It can fall into certain pitfalls such as the total number of recommendations being below the total number of users in a group, making it so the last users do not have a say in the resulting list. This algorithm was not in the original set, and was implemented for this paper.

Borda Count Algorithm (BDC) [3] assigns each item a point value based on their ranking in each user's preference order. The item ranked first by a user receives a certain number of points. The item ranked second receives one fewer point, and so on, with the last-ranked item receiving zero points. After all the users have submitted their rankings and the points are tallied, the top items with the highest total number of points are selected. The Borda Count is a form of preferential voting that aims to determine the item who is preferred by the most users overall, rather than simply the item who

receives the most first-place ranks. This algorithm was not in the original set, and was implemented for this paper.

Average Without Misery Algorithm (AVGNM) [3] averages individual ratings, after excluding items with individual ratings below a certain given threshold (in our case, a rating of 1.0). A list of allowed items (items above the threshold) is created before averaging all resulting item ratings and collecting the total amount of recommendations. This algorithm was not in the original set, and was implemented for this paper.

The resulting 10 items for each algorithm were saved and compiled into a file. This process was repeated for each top-k restricted file, and certain algorithms performed differently depending on how many user-item pairings they had access to.

2.4 Evaluation

The evaluations were divided into two types: coupled and decoupled evaluation [8]. Coupled and decoupled evaluation had binarized feedback set with a positive threshold of 4.

The **coupled evaluation** [8] considered the original ML-1M file to be its ground-truth for all iterations. This means the predicted values for certain user-item pairings are unknown to the evaluation, regardless of top-k restrictions. Effectively, the evaluation checks performance of the overall solution considering withheld or unknown information.

The **decoupled evaluation** [8] considered each topk-restricted file to be the ground-truth in each iteration. This simulates a scenario where all user preferences are known to the system (where the universe of preferences is the top-k ratings), and only evaluates the GRS's ability to combine those preferences.

Both evaluations were done with all top-k restricted files iteratively, and results were saved separately. Only the Topk-1 file had a feedback polarity debiasing of -3. Final results were separated by algorithm, aggregating values among the various group compositions with different approaches (mean, min and minmax).

2.5 Metrics

Six commonly used metrics were employed to check the performance of each algorithm: Normalized Discounted Cumulative Gain (NDCG), Discounted Cumulative Gain (DCG), Recall, Bounded Recall (works similarly to Recall, with the only difference being that the denominator checks for a minimum between correct items and recommended items, instead of just using correct items in the "normal" Recall), Discounted First Hit (DFH), and Mean Reciprocal Rank (MRR).

3. Results

All top-k variations and their nine algorithms, both coupled and decoupled, were checked with the six different metrics. Most notable are the variations between very small top-k numbers such as Fig. 3 and Fig. 4, and bigger top-k values such as Fig. 5 and Fig. 6. They were chosen as they demonstrated the variation between smaller, medium and larger values for top-k the best.

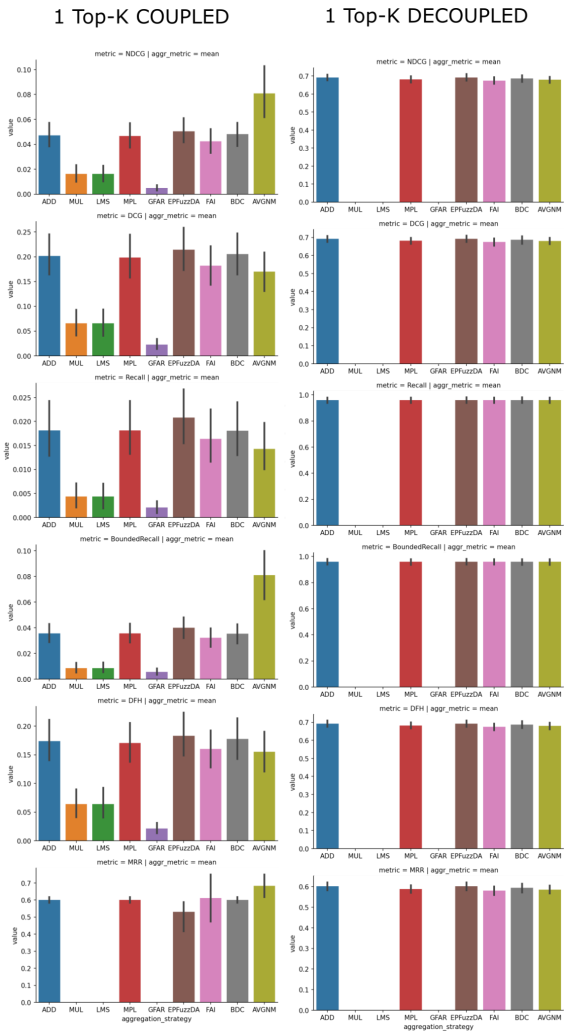

Fig. 3 - Coupled and decoupled results for dataset with only the top 0.05% (1) item that each user rated.

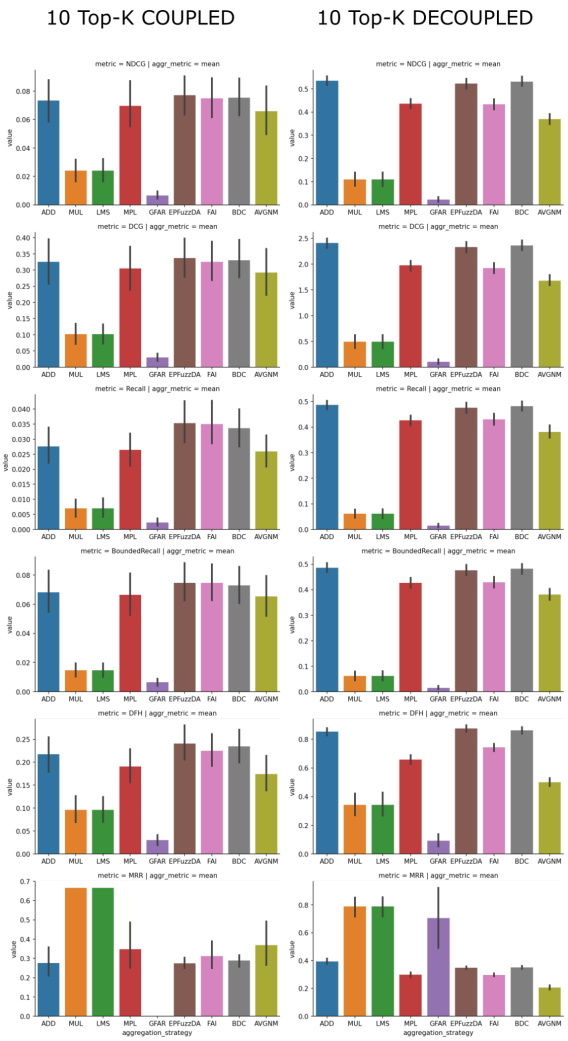

Fig. 4 - Coupled and decoupled results for dataset with only the top 0.5% (10) items that each user rated.

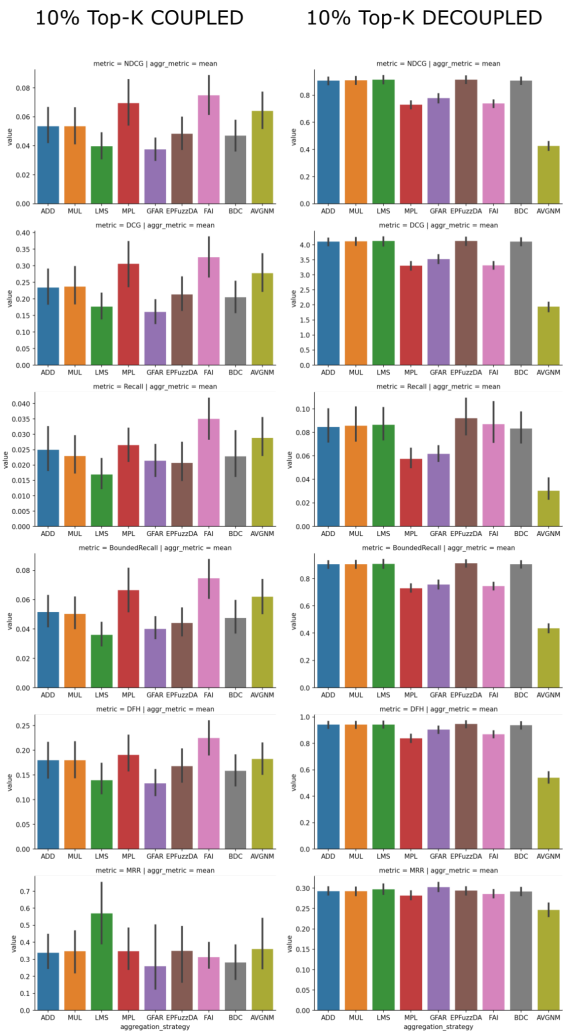

Fig. 5 - Coupled and decoupled results for dataset with only the top 10% (202) items that each user rated.

It is important to note that, since the total number of recommendations is ten, some top-k restrictions such as Fig. 3 are working with less user preference data than the algorithm will eventually recommend. Top-k restrictions such as in Fig. 4 are working with exactly the same amount of data that the algorithm will eventually recommend (ten user-item pairs). This has a significant impact on final results, especially with certain algorithms that depend on user-item pairings more.

As can be observed in Fig. 3 and Fig. 4, when the top-k value is lower than the total expected recommendations, the AVGNM algorithm can recommend less than the expected number of items, regardless of having the AVGNM threshold done before or after the selection of available items. This happens because there are not enough items in the top-k restricted file that are above the threshold value set. Evidently, in Fig. 5 and Fig. 6, this noted phenomenon does not occur when there are enough items in the top-k restriction.

In the case of comparing coupled and decoupled values, the decoupled graphs of most top-k restrictions tend to have higher values for their metrics. This is because, as explained earlier in section 2.4, the decoupled method takes the topk-restricted file to be the ground-truth of the matter.

To see figures of all top-k variations, please visit the folder "/eval_img_results/" in the github repository: (https://github.com/lpeska/GRS_eval_props).

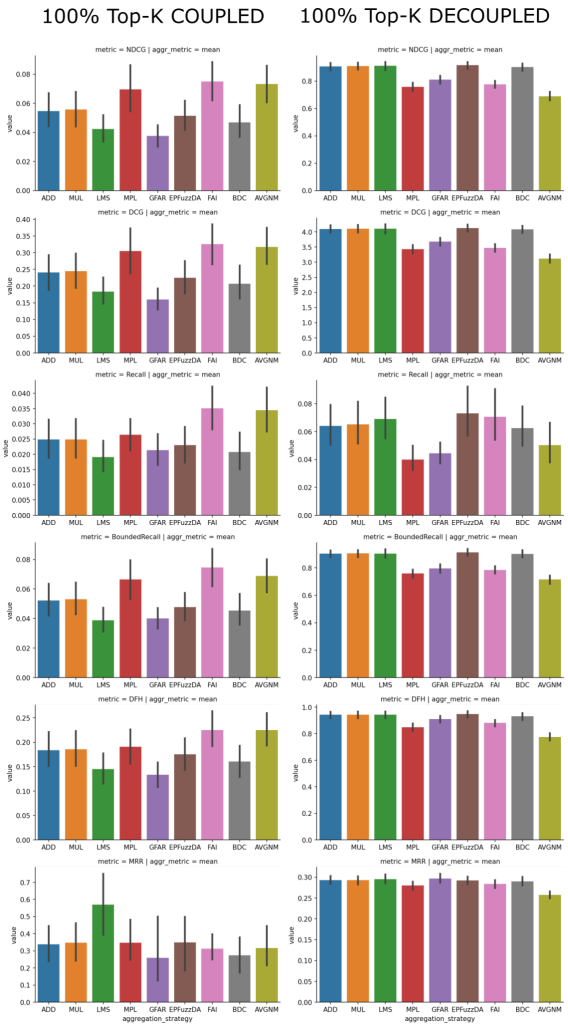

Fig. 6 - Coupled and decoupled results for dataset with only the top 100% (2020) items that each user rated.

4. Conclusion

While lower top-k values favoured algorithms such as MPL and EPFuzzDA, higher top-k values favoured algorithms such as MUL and LMS. This is in accordance with the current understanding of the EPFuzzDA algorithm performing better with harsher top-k restrictions and its list-wise approach [9].

When using top-k restrictions that were below the total number of expected recommendations, the MUL, LMS, and GFAR algorithms could not offer significant results. Algorithms such as MPL, FAI and AVGNM remained relatively consistent regardless of top-k restrictions.

These results, when compiled, could be skewed by different definitions of 'fairness' in regards to evaluation. Regardless, the top-k restriction has a significant impact on the performance of a given GRS algorithm, considering the metrics observed.

5. References

[1] Konstan, J. A., & Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User modeling and user-adapted interaction*, *22*, 101-123.

[2] Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender systems handbook*, 257-297.

[3] Masthoff, J. (2010). Group recommender systems: Combining individual models. In *Recommender systems handbook* (pp. 677-702). Boston, MA: Springer US.

[4] Sacharidis, D. (2019, April). Top-n group recommendations with fairness. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing* (pp. 1663-1670).

[5] Kaya, M., Bridge, D., & Tintarev, N. (2020, September). Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In *Proceedings of the 14th ACM Conference on Recommender Systems* (pp. 101-110).

[6] Zhao, Q., Chang, S., Harper, F. M., & Konstan, J. A. (2016, September). Gaze prediction for recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 131-138).

[7] Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., & Gay, G. (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, *25*(2), 7-es.

[8] Peska, L., & Malecek, L. (2021). Coupled or Decoupled Evaluation for Group Recommendation Methods?. In *Perspectives@ RecSys*.

[9] Malecek, L., & Peska, L. (2021, June). Fairness-preserving group recommendations with user weighting. In *Adjunct Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization* (pp. 4-9).hani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. *Recommender systems handbook*, 257-297.