# Fairness in Group Recommendation - Third Assignment

Author: Alberto Tontoni

Project Source Code: https://github.com/tontonialberto/advanced-topics-cs

## Table of Contents

# Introduction

The aim of this paper is to describe and summarize the work done for the *Advanced Topics in Computer Science* course held at Third University of Rome, regarding the lectures on Fairness in Group Recommendations given by Professor Stefanidis, K.

The proposed assignments are about designing, implementing, and evaluating a Recommendation System (RS) capable of making Sequential Group Recommendations using the User-Based Collaborative Filtering approach. To do this, an individual user recommender is designed and implemented: since a key factor to make good recommendations is to use an appropriate similarity metric between users, several alternatives are discussed. Namely, the Pearson Correlation Coefficient is compared with the Jaccard similarity and the Improved Triangle Similarity complemented with User Rating Preferences (ITR).

The next step is to extend our RS in order to make recommendations to group of users, taking into account the opinion of all members whilst making good recommendations for the group. The proposed method aggregates the preferences of individual group members so as to obtain a "fake" user who represents the group, and uses the individual user RS to make predictions for the group. We discuss the implications that an aggregation method has on the fairness of the RS with respect of two commonly used methods: the Average and the Least Misery aggregation methods. We also propose a method that takes the group disagreement into account.

Lastly, the group RS is furtherly extended so as to consider many interactions between a group of users and the system itself. The proposed Sequential Group Recommendation System is a slight variation of the Sequential Dynamic Adaptation Aggregation method [3], which dinamically balances between Average and Least Misery Aggregation using previous disagreement information.

The discussed concepts are implemented using the Python programming language. For the purpose of evaluation, the MovieLens 100K small dataset has been used. It contains ratings given by 610 users to 9743 movies.

This paper is organized as follows: the first three sections respectively summarize the purpose and the proposed design for User-Based Collaborative Filtering, Group Recommendation, and Sequential Group Recommendation. The last two sections describe the approach used for evaluation and discuss the obtained results.

# User-Based Collaborative Filtering

The main aim of a Recommendation System is to find the items which are more likely to match the preferences of a user (or a group of users). The two main approaches for creating such a system are *Content-Based Filtering* and *Collaborative Filtering*. In *Content-Based Filtering*, the features (ie. the "content") of the items are taken into account to provide the best recommendations for a given user. *Collaborative Filtering* instead focuses only on the interactions between users and items (ie. the rating given by a user to an item).

Collaborative Filtering can be further divided into two methodologies: *User-Based* and *Item-Based*. In the *User-Based* method, items are recommended by looking at similarities between users: this means that if user A is highly similar to user B, recommendations for user A are likely to include items positively rated by user B. In the *Item-Based* method, recommendations are made by looking for items which are similar to the ones liked by the considered user.

Here is proposed a method to design a *User-Based Collaborative Filtering* RS which takes into account software systems qualities like performance and testability.

The strongest assumption that has been made is that the given dataset is fixed ie. it doesn't change over time. Access to a read-only dataset can be optimized in many ways: for example, precomputing and caching the values that are accessed most often. This strategy is extensively used in this project to reduce the time cost of heavy computations, like the user similarity matrix.

Another assumption is that the dataset is somehow "clean": for example, there is at most 1 rating for an item by the same user. This simplification has been made so as to avoid cleaning the dataset.

A User-Based CF Recommendation System is basically made of three main software elements:

- **Similarity Calculator**: computes similarity between users. Many similarity functions do exist. The simplest ones take only into account the number of commonly rated items, whereas more complex functions also consider the rating differences between common items, or even the items not in common between two users.
- **Predictor**: uses similarity to provide an estimate for an item not rated by a user.
- **Recommender**: recommends the items with the highest predictions to a single user who has not rated those items.

It is clear that every element depends on the previous ones. This guides a simple yet useful architectural decomposition of our system.

## Similarity Metrics

Here are briefly presented some alternatives for similarity computation.

#### Pearson Correlation Coefficient

This is the first similarity measure that has been implemented in our project. It measures the similarity between two users $u, v$ and it's defined as follows:

$$PCC(u,v) = \frac{\sum_{i \in P}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in P}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{i \in P}(r_{v,i} - \overline{r}_v)^2}}$$

where:

- $P$ is the set of items rated by both $u$ and $v$;
- $r_{u,i}$ is the rating of user $u$ for item $i$;
- $\overline{r}_u$ is the mean of the ratings of user $u$.

The similarity interval is $[-1, 1]$. An edge case to take into account in the implementation is when $P = \emptyset$ ie. the two users have no commonly rated items: this indicates no correlation between users, henceforth $PCC(u, v)$ is zero in that case.

#### Jaccard Similarity

The Jaccard similarity is one of the simplest similarity measures. It's defined as the cardinality of the intersection divided by the cardinality of the union of the sets of items rated by two users:

$$J(u, v) = \frac{|u \cap v|}{|u \cup v|}$$

The similarity interval is $[0, 1]$. The similarity is maximum (ie. 1) when $u, v$ have rated exactly the same items, and reaches its minimum (ie. 0) when $u, v$ have no commonly rated items.

Note that it doesn't take into account the actual ratings: this intuitively tells us that the predictions made using the Jaccard index may be poor (since it uses a very restricted set of information).

#### ITR Similarity

The ITR similarity is defined as the product of other two similarities: the *Improved Triangle Similarity* and the *User Rating Preferences*. Therefore it's defined as follows:

$$sim^{ITR}(u, v) = sim^{TRIANGLE'}(u, v) * sim^{URP}(u, v)$$

The *Improved Triangle Similarity* is defined as follows:

$$sim^{TRIANGLE'}(u, v) = 1 - \frac{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} + \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

where $I_{uv}$ is the set of items rated by **either** $u$ or $v$. If $u$ does not rate a particular item $i$, then its rating will be zero. Therefore, this similarity takes into account also items not in common between the two users.

For the *User Ratings Preferences*, the following formula has been employed:

$$sim^{URP}(u, v) = 1 - \frac{1}{1 + +exp(-|\overline{r}_u - \overline{r}_v| * |\sigma_u - \sigma_v|)}$$

where:

- $\overline{r}_u$ is the average rating of user $u$ over the set of items rated either by $u$ or $v$;
- $\sigma_u$ is the standard deviation of user $u$ ratings, using the above mentioned average and dividing by the number of items rated by $u$:

$$\sigma_u = \sqrt{\frac{\sum_{i \in I_u} (r_{ui} - \overline{r_u})^2}{|I_u|}}$$

**Important note**: the implemented version for *User Rating Preferences* uses a slightly different computation for $\sigma_u$ with respect to the formula presented in [1] because I've empirically noticed that prediction results are better (*with respect to the specific experiment I've made*). More specifically:

- The formula as presented above performs worse than Pearson Correlation;
- The tweaked formula performs better than Pearson Correlation;
- By the way, both versions perform better than Jaccard similarity.

These results will be further explained in the Results section. However, the obtained results cannot be said to be generally better than those obtained in [1] as it would need way more complex evaluations.

## Prediction Function (Mean-Centered Aggregation)

In general, a prediction function $pred : U \times I \rightarrow R$ takes as input a user $u$ and an item $i$ not rated by the user, and returns an estimate of the rating that $u$ would give to $i$ (or, more generally speaking, a *score* for that item).

The similarity function seen during classes is the following:

$$pred(u,i) = \overline{r}_u + \frac{\sum_{v \in N} sim(u,v) * (r_{v,i} - \overline{r}_v)}{\sum_{v \in N} sim(u,v)}$$

where:

- $\overline{r}_u$ is the average rating of user $u$;
- $N$ is the set of users who commonly rate item $i$. We can choose to restrict this set to the most similar users with respect to $u$;
- $sim(u,v)$ is an arbitrary similarity function.

As seen in Assignment 1, this prediction function returns ratings which are highly above or below the range 1-5 for this dataset.

Another prediction function has been considered, which is the *Mean-Centered Aggregation* defined as follows:

$$pred(u,i) = \overline{r}_u + \frac{\sum_{v \in N} sim(u,v) * (r_{v,i} - \overline{r}_v)}{\sum_{v \in N} |sim(u,v)|}$$

The only difference with the previous formula is in the absolute value of the similarities at the denominator.

# Group Recommendation

The basic idea behind Group Recommendation is to provide an estimate for the most relevant items for a given group of users. In our example, it is required to provide the k-top movies for a group.

The proposed solution is to aggregate the individual group members' preferences to compute a "fake" user which represents the whole group, and to choose the top-k items from the preferences of this new user.

For the sake of simplicity, the following assumptions have been made:

- The length of the top-k items list is equal to 10 (ie. k=10);
- The group does not change over time: members are not allowed to leave the group once it's been formed, and external users are not allowed to join an already formed group.

A Group Recommendation System can be decomposed in the following software elements:

- **Group Predictor**: its main responsibility is to aggregate group members' individual preferences for an item to provide a group preference for that item.

  Aggregation is based on relevance. The *relevance* $rel(u,i)$ of item $i$ for user $u$ is either the rating $r_{u,i}$, or the prediction $pred(u,i)$ if $u$ has not rated $i$. This approach estimates individual user preferences by using the **Predictor** module presented in the previous sections.

- **Group Recommender**: selects the top-k recommendations for the group using the aggregated group preferences.

There are many ways to aggregate individual preferences to obtain group preferences. In the following sections, two well-enstablished methods will be discussed (Average and Least Misery Method). An additional method will also be considered to take into account group members' disagreements.

#### Average Aggregation Method

The Average Aggregation formula for group $G$ on item $i$ is simply the average of the individual relevances of each group member, and it's defined as follows:

$$rel_{AVG}(G,i) = \frac{\sum_{u \in G} rel(u,i)}{|G|}$$

This formula is expected to work well when group members share similar preferences. It however might recommend only the items preferred by the majority of the group, thus penalizing users who do have different tastes in the same group.

#### Least Misery Aggregation Method

The Least Misery Aggregation formula for group $G$ on item $i$ takes only into account the lowest individual relevance given by a group member:

$$rel_{LM}(G,i) = min_{uinG}(rel(u,i))$$

In this formula, it is said that a user in the group acts as a veto with respect to the group relevance. This formula may be good to avoid penalizing always the same subset of group members. However, it might lead to recommendations which aren't particularly liked by any user.

**Average Pair-wise Disagreements**

The above mentioned methods do not consider whether the group members agree on a recommended item. A way to formalize disagreement [2] is *the degree of consensus in the relevance scores for an item $i$ among members of a group $G$*.

The metric chosen to quantify disagreements is the *Average Pair-wise Disagreements* function, defined as follows:

$$dis(G,i) = \frac{2}{|G|(|G|-1)} \sum_{u,v \in G \wedge u=v} (|rel(u,i) - rel(v,i)|)$$

Intuitively, the higher the consensus among group members on item $i$, the lower the disagreement.

**Consensus Aggregation Method**

Proceeding with the approach proposed in [2], the chosen method to make group recommendations considering also the degree of disagreement among group members is by using a *Consensus* function, defined as follows:

$$f(G,i) = w_1 \cdot rel(G,i) + w_2 \cdot (1 - dis(G,i))$$

where:

- $w_1, w_2$ are constant weights, $w_1 + w_2 = 1$;
- $rel(G,i)$ is an arbitrary group aggregation function for group $G$ on item $i$;
- $dis(G,i)$ is an arbitrary disagreement function for group $G$ on item $i$.

The *Consensus* function is simply a weighted sum of an underlying **Group Predictor** and the disagreement term. It is easy to verify that for $w_1 = 1$ the formula falls back to the underlying group aggregation function. As $w_2$ increases, however, items with an high degree of disagreement are penalized.

## Sequential Group Recommendation

The idea of Group Recommendation presented in the previous sections implicitly assumes that every interaction between the group and the recommender is completely independent from the others. However, real-world scenarios may require to relate interactions from the same group for many reasons: for instance, to avoid recommending the same item twice to the same group. Another reason is to leverage information from previous iterations to improve the recommender's fairness.

This leads us to the idea of Sequential Group Recommendation. Here, instead of treating a single recommendation, a series of recommendations is considered. The reasoning behind this approach is to consider the so-called satisfaction of individual group members at a given iteration and, if a user is not currently satisfied, he/she has either been satisfied in a previous iteration or will be satisfied in the next one.

The rest of this section proposes an architecture for a simple Sequential Group Recommendation System. The following assumptions (including the previously mentioned ones) have been made:

- An item cannot be recommended twice to the same group at different iterations;
- For simplicity, the same item may be recommended twice in the case of *individual* users recommendations.

The Sequential Group RS can be built on top of the software elements described in the previous sections. The system can be decomposed to a few elements:

- **User Satisfaction Metric**: tells how much a group recommendation reflects the preferences of an individual group member.
- **Sequential Predictor**: similarly to the Group Predictor, it estimates the score of an item for a group of users. The main difference is that previous iterations have to be considered. As will be explained, a sequential prediction can leverage user satisfaction information to try to satisfy different users in different iterations.
- **Sequential Group Recommender**: makes sequential predictions on all items for a group, and produces an ordered list of top-k items with the highest scores for the group.

The individual satisfaction of user $u$, with respect to the top-k recommendations $Gr_j$ produced at iteration $j$ by a group recommender, can be defined as follows:

$$sat(u, Gr_j) = \frac{\sum_{d_z \in Gr_j} rel(u, d_z)}{\sum_{d_z \in A_{u,j}} rel(u, d_z)}$$

where:

- $rel(u, d_z)$ is the previously defined relevance of user $u$ on item $d_z$;
- $A_{u,j}$ is the individual recommendation list of top-k items for user $u$.

High satisfaction values (ie. close to 1) indicate that the group recommendation is close to the ideal case of the single user recommendation, whereas low values (ie. close to 0) indicate the opposite.

> An implementation detail is that satisfaction values may be outside the [0,1] interval if the prediction function returns values outside the range of the actual ratings.

Similarly, the overall individual satisfaction of user $u$ across $N$ iterations is simply the average satisfaction of the single iterations:

$$satO(u, GR) = \frac{\sum_{j=1}^{N} sat(u, Gr_j)}{N}$$

The sequential predictor proposed in this project is a slight variation of the Sequential Dynamic Adaptation Aggregation (SDAA) method proposed in [3]:

$$rel_{SEQ}(G, d_z, j) = (1 - \alpha_j) \cdot rel_{AVG}(G, d_z) + \alpha_j \cdot rel_{LM}(G, d_z)$$

where $rel_{AVG}$ and $rel_{LM}$ are respectively the Average and Least Misery aggregation methods used for Group Recommendations.

Moreover, $\alpha_j$ is a value between 0 and 1 that is dinamically updated between iterations and indicates the degree of disagreement between group members.

At iteration $j$, the value for $\alpha_j$ is calculated as follows:

$$\alpha_j = max_{i \in \{j-k,\dots,j-1\}} [max_{u \in G} sat(u, Gr_i) - min_{u \in G} sat(u, Gr_i)]$$

where $k \geq 1$ is an hyperparameter that indicates how many previous iterations must be considered in the computation of disagreement. For $j < k$, the aggregation simply falls back to the average case. This formula is a slight variation of the one proposed in [3] in that it considers only the iteration with the highest disagreement (among the most recent ones).

The working principle of the proposed aggregation method can be summarized as follows:

- At the beginning (ie. first iterations), all the group members are considered equal, henceforth predictions are made using the Average method;
- Whenever the disagreement rises, so does $\alpha_j$, leading to more inclusive predictions that resemble the Least Misery approach;
- The value for $\alpha_j$ is updated at each iteration, so as to automatically find the right balance between the two methods whose advantages and drawbacks have been presented in previous sections.

# Evaluation

This section describes the approach used to evaluate the Recommendation System and the discussed alternatives.

## User-Based Collaborative Filtering Evaluation

To evaluate which similarity measure leads to better prediction, the following experiment will be conducted:

- Instantiate three recommenders $RS_{PCC}, RS_{JACCARD}, RS_{ITR}$. Both use the same prediction function but a different similarity function;
- Select all items rated by a user: for each item, do a prediction with each of the recommenders and compare the results.

The following measures of error will be used:

- Absolute Error: for each item in the dataset, the difference between the true rating and the predicted rating;
- Mean Absolute Error (MAE);
- Score: for each item, increase the counter of the predictor whose prediction led to the minimum absolute error for that item.

The best predictor is then chosen according to its MAE and score values.

## Group Recommendation Evaluation

For the case of Group Recommendation, a group of three users is chosen and top-10 movies recommendations are made using the presented aggregation methods. We then make some considerations on how the group recommendation items change according to the aggregation method, considering also the individuals' preferences and group disagreements.

## Sequential Group Recommendation Evaluation

Lastly, the proposed Sequential Group Recommender is compared with aggregation methods which only consider single iterations (ie. the Average and Least Misery methods used for Group Recommendations). The evaluation is done over 5 iterations on a group of 3 users, 2 similar and 1 dissimilar from the others. Their similarity matrix according to the Pearson Correlation Coefficient is shown below:

| | **User 599** | **User 382** | **User 361** |
|---|---|---|---|
| **User 599** | 1 | 0.73 | -0.42 |
| **User 382** | 0.73 | 1 | -0.35 |
| **User 361** | -0.42 | -0.35 | 1 |

# Results

## User-Based Collaborative Filtering Results

Unless otherwise specified, all the shown experiments use the *Mean-Centered Aggregation* as prediction function on all neighbors.

Here is the difference of scores between the ITR formula as presented in [1] and the tweaked version. The experiment is conducted on user with Id 1:

| Predictor | Score | Mean Absolute Error |
|---|---|---|
| itr | 61 | 0.451366 |
| tweaked itr | 122 | 0.334832 |

Here is the output of the evaluation conducted on user with Id 1 (only few predictions are shown):

| Item | True Rating | Pred. (pearson) | Pred. (tweaked itr) | Pred. (jaccard) | Abs. Error (pearson) | Abs. Error (tweaked itr) | Abs. Error (jaccard) | Best |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 4.40474 | 4.56465 | 4.64696 | 0.404739 | 0.564655 | 0.646957 | pearson |
| 3 | 4 | 4.07405 | 4.20467 | 4.08122 | 0.0740497 | 0.204666 | 0.0812244 | pearson |
| 6 | 4 | 4.39643 | 4.46715 | 4.65416 | 0.396434 | 0.467154 | 0.654157 | pearson |
| 47 | 5 | 4.84033 | 4.57481 | 4.79007 | 0.159666 | 0.425195 | 0.209931 | pearson |
| 50 | 5 | 4.90937 | 5.00488 | 5.04639 | 0.0906338 | 0.00487751 | 0.0463876 | tweaked itr |
| 70 | 3 | 3.9321 | 3.79314 | 4.11133 | 0.9321 | 0.793139 | 1.11133 | tweaked itr |
| 101 | 5 | 5.1171 | 4.90185 | 4.98159 | 0.117096 | 0.0981468 | 0.0184127 | jaccard |
| 110 | 4 | 4.44158 | 4.5533 | 4.75535 | 0.441582 | 0.553296 | 0.755348 | pearson |

Here's the result of the evaluation conducted on user with Id 1, using the three recommenders:

| Predictor | Score | Mean Absolute Error |
|---|---|---|
| pearson | 73 | 0.393644 |
| tweaked itr | 122 | 0.334832 |
| jaccard | 37 | 0.434312 |

It means that on this experiment, the similarity which leads to the best recommendations is the tweaked ITR version, since it has the highest score and lowest MAE.

## Group Recommendation Results

Unless otherwise specified, all the shown esperiments use the *Mean-Centered Aggregation* as prediction function on the 50 most similar neighbors. The similarity between users is computed using the *Pearson Correlation Coefficient*.

**Average Aggregation Results**

Here are shown the top-10 recommendations for group of users [71, 390, 467] using the Average Aggregation:

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 1 | 1194 | 5.22314 | 3.6 | 6.27132 | 5.79811 | 1.78088 |
| 2 | 3552 | 5.18821 | 4.34074 | 5.42579 | 5.79811 | 0.971577 |
| 3 | 1237 | 5.1759 | 4.85455 | 5.13362 | 5.53953 | 0.456653 |
| 4 | 475 | 4.99091 | 4.96364 | 5.55379 | 4.45532 | 0.732317 |
| 5 | 58 | 4.96766 | 4.88004 | 5.02294 | 4.19285 | 0.0952647 |
| 6 | 1663 | 4.88981 | 3.6 | 6.27132 | 4.79811 | 1.78088 |
| 7 | 1204 | 4.87472 | 5.51429 | 3.9997 | 5.11018 | 1.00972 |
| 8 | 280 | 4.83674 | 5.11111 | 4.30534 | 5.09378 | 0.537182 |
| 9 | 3266 | 4.80001 | 6.35455 | 3.83951 | 4.20597 | 1.67669 |
| 10 | 104374 | 4.74037 | 4.90195 | 4.18326 | 5.1359 | 0.635093 |

We can see that item 1194 and item 1663 have high disagreement values. By looking at the individual relevances, it seems that user 71 is penalized to some extent.

**Least Misery Aggregation Results**

Here are shown the top-10 recommendations for group of users [71, 390, 467]:

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 1 | 58 | 4.88004 | 4.88004 | 5.02294 | 4.19285 | 0.0952647 |
| 2 | 1237 | 4.85455 | 4.85455 | 5.13362 | 5.53953 | 0.456653 |
| 3 | 475 | 4.45532 | 4.96364 | 5.55379 | 4.45532 | 0.732317 |
| 4 | 3552 | 4.34074 | 4.34074 | 5.42579 | 5.79811 | 0.971577 |
| 5 | 223 | 4.33915 | 4.55542 | 4.33915 | 4.66791 | 0.219174 |
| 6 | 111 | 4.3201 | 4.37265 | 4.3201 | 4.48501 | 0.109942 |
| 7 | 280 | 4.30534 | 5.11111 | 4.30534 | 5.09378 | 0.537182 |
| 8 | 364 | 4.23196 | 4.23196 | 4.40791 | 4.33987 | 0.117303 |

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 9 | 47099 | 4.22077 | 4.45328 | 4.22077 | 4.25983 | 0.155004 |
| 10 | 104374 | 4.18326 | 4.90195 | 4.18326 | 5.1359 | 0.635093 |

We can see that each recommendation only depends on the lowest relevance of a single user. Disagreements also seem to have reduced compared with the Average Aggregation scenario.

**Consensus based on Average Aggregation Results**

Here are performed two experiments for group recommendation using the *Consensus* function, which employs an underlying Average Aggregation function for group aggregation, and uses Average Pairwise Disagreements as disagreement function.

The two experiments show how the recommended items have lower disagreement scores as the hyperparameter $w_2$ increases.

Here are shown the top-10 results for group of users [71, 390, 467] for $w_1 = 0.8, w_2 = 0.2$:

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 1 | 1237 | 4.24939 | 4.85455 | 5.13362 | 5.53953 | 0.456653 |
| 2 | 3552 | 4.15625 | 4.34074 | 5.42579 | 5.79811 | 0.971577 |
| 3 | 58 | 4.15507 | 4.88004 | 5.02294 | 4.19285 | 0.0952647 |
| 4 | 475 | 4.04627 | 4.96364 | 5.55379 | 4.45532 | 0.732317 |
| 5 | 1194 | 4.02234 | 3.6 | 6.27132 | 5.79811 | 1.78088 |
| 6 | 280 | 3.96196 | 5.11111 | 4.30534 | 5.09378 | 0.537182 |
| 7 | 1204 | 3.89783 | 5.51429 | 3.9997 | 5.11018 | 1.00972 |
| 8 | 104374 | 3.86527 | 4.90195 | 4.18326 | 5.1359 | 0.635093 |
| 9 | 223 | 3.77283 | 4.55542 | 4.33915 | 4.66791 | 0.219174 |
| 10 | 112 | 3.76063 | 4.80263 | 4.01486 | 5.05492 | 0.693377 |

This result is similar to the Average Aggregation situation (as $w_2$ is low), even though some items with high disagreement have been slightly penalized.

Here are shown the top-10 results for group of users [71, 390, 467] for $w_1 = 0.6, w_2 = 0.4$:

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 1 | 58 | 3.34249 | 4.88004 | 5.02294 | 4.19285 | 0.0952647 |
| 2 | 1237 | 3.32288 | 4.85455 | 5.13362 | 5.53953 | 0.456653 |
| 3 | 3552 | 3.1243 | 4.34074 | 5.42579 | 5.79811 | 0.971577 |

| # | Item | Pred. (Group) | Pred. (User 71) | Pred. (User 390) | Pred. (User 467) | Disagreement |
|---|---|---|---|---|---|---|
| 4 | 475 | 3.10162 | 4.96364 | 5.55379 | 4.45532 | 0.732317 |
| 5 | 280 | 3.08717 | 5.11111 | 4.30534 | 5.09378 | 0.537182 |
| 6 | 223 | 3.02483 | 4.55542 | 4.33915 | 4.66791 | 0.219174 |
| 7 | 111 | 2.99158 | 4.37265 | 4.3201 | 4.48501 | 0.109942 |
| 8 | 104374 | 2.99018 | 4.90195 | 4.18326 | 5.1359 | 0.635093 |
| 9 | 364 | 2.94903 | 4.23196 | 4.40791 | 4.33987 | 0.117303 |
| 10 | 47099 | 2.92477 | 4.45328 | 4.22077 | 4.25983 | 0.155004 |

Here we can see that the vast majority of top-10 items have very low disagreement values. Moreover, even though the prediction score for the group is low, this is not related to the individual relevances, which are relatively high with respect to the 1-5 range of movie ratings.

## Sequential Group Recommendation Results

Unless otherwise specified, all the shown esperiments use the *Mean-Centered Aggregation* as prediction function on the 20 most similar neighbors. The similarity between users is computed using the *Pearson Correlation Coefficient*.

The first two tables show the individual users satisfactions for Group Recommendation across five iterations, using respectively the Average and the Least Misery aggregation methods. The third table uses the proposed variation of the SDAA method employed in Sequential Group Recommendation. The value for $k$ has been set to 2, which means that $\alpha_j$ will be the maximum disagreement among the last 2 iterations.

For each iteration, the top-10 recommendations are also reported.

**Average Aggregation over 5 iterations**

As expected, the average method produces recommendations more aligned to the preferences of the majority, which results in higher satisfactions for similar users (ie. 599 and 382) and lower values for the outlier (which is user 361).

| Iteration | User 599 Satisfaction | User 382 Satisfaction | User 361 Satisfaction | Group Recommendations |
|---|---|---|---|---|
| 1 | 1.05125 | 0.908344 | 0.788968 | [750, 4226, 1673, 1376, 4973, 296, 1196, 2959, 3703, 1214] |
| 2 | 0.877891 | 0.922988 | 0.766436 | [225, 60069, 122882, 2571, 1206, 1249, 58559, 107, 555, 2542] |
| 3 | 0.894037 | 0.915297 | 0.666431 | [105504, 44555, 64957, 1148, 2640, 1250, 54001, 912, 1204, 1230] |

| Iteration | User 599 Satisfaction | User 382 Satisfaction | User 361 Satisfaction | Group Recommendations |
|---|---|---|---|---|
| 4 | 0.897665 | 0.800767 | 0.757344 | [1265, 2019, 1198, 1223, 1136, 81847, 68157, 1225, 7387, 4963] |
| 5 | 0.883496 | 0.827584 | 0.710057 | [260, 318, 80463, 27156, 1237, 1639, 33493, 1967, 3275, 4011] |

**Least Misery Aggregation over 5 iterations**

The Least Misery method is instead more inclusive with respect to the outlier. In fact, its satisfaction is higher at the cost of lower satisfaction scores for other users.

| Iteration | User 599 Satisfaction | User 382 Satisfaction | User 361 Satisfaction | Group Recommendations |
|---|---|---|---|---|
| 1 | 0.935791 | 0.856227 | 0.850821 | [750, 4226, 4973, 1673, 60069, 1225, 122882, 3168, 107, 81847] |
| 2 | 0.934457 | 0.645637 | 0.825109 | [1136, 1223, 2161, 2692, 2700, 3996, 4963, 5995, 109374, 80463] |
| 3 | 0.833805 | 0.72288 | 0.77052 | [119145, 27156, 10, 17, 163, 231, 356, 466, 555, 589] |
| 4 | 0.794028 | 0.764885 | 0.817752 | [608, 1198, 1252, 1356, 1376, 1393, 1517, 1639, 2087, 68157] |
| 5 | 0.816395 | 0.692475 | 0.769118 | [4085, 136562, 142488, 176371, 80489, 58559, 1221, 1953, 4848, 2791] |

**Variation of SDAA over 5 iterations (k=2)**

Lastly the sequential method - which actually considers previous iterations - produces results inbetween the aforementioned approaches: it starts in the same way as the Average Aggregation and uses the disagreement of the most recent iterations as a feedback to give more importance to unsatisfied user(s).

| Iteration | User 599 Satisfaction | User 382 Satisfaction | User 361 Satisfaction | Group Recommendations |
|---|---|---|---|---|
| 1 | 1.05125 | 0.908344 | 0.788968 | [750, 4226, 1673, 1376, 4973, 296, 1196, 2959, 3703, 1214] |
| 2 | 0.877891 | 0.922988 | 0.766436 | [225, 60069, 122882, 2571, 1206, 1249, 58559, 107, 555, 2542] |
| 3 | 0.8657 | 0.797454 | 0.787049 | [1225, 81847, 54001, 1198, 1223, 1136, 3168, 68157, 1250, 4963] |
| 4 | 0.914819 | 0.913825 | 0.64819 | [105504, 44555, 64957, 1148, 2640, 912, 1204, 1230, 1265, 2019] |

| Iteration | User 599 Satisfaction | User 382 Satisfaction | User 361 Satisfaction | Group Recommendations |
|---|---|---|---|---|
| 5 | 0.934457 | 0.684655 | 0.80144 | [80463, 27156, 1639, 2692, 119145, 589, 6, 3996, 7387, 608] |

# References

[1] Fkih F., *Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison*, Journal of King Saud University - Computer and Information Sciences, 2022

[2] Amer-Yahia S., Basu Roy S., Chawla A., Das G., Yu C., *Group Recommendation: Semantics and Efficiency*, Proceedings of the VLDB Endowment, 2009

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