Fairness in Group Recommendations - Assignment 2

Author: Alberto Tontoni

This report describes the implementation of a basic Group Recommendation System for the MovieLens 100K small dataset, leveraging the User-based Collaborative Filtering approach implemented in Assignment 1.

Table of Contents

Design and Implementation of 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.

The rest of this section describes the approach in detail.

Assumptions

For the sake of simplicity, and to stay consistent with the previous assignment work, 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;
- The software elements presented in Assignment 1 (**Predictor** and **Similarity**) have been already implemented, as they will be used to produce group recommendations.

Design

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. It is clear that this approach requires a way to compute user predictions (ie. the **Predictor** module implemented in the previous assignment).

- **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.

A method for counting disagreements (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 [1] 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.

Aggregation Method that considers disagreement

Proceeding with the approach proposed in [1], 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.

Evaluation

Here is described the approach used to evaluate the system:

- Choose a group of three users;
- Perform a top-10 movies recommendation on that group using different aggregation functions;
- Make some considerations on how the group recommendation items change according to the aggregation method, considering also the individuals' preferences and group disagreements.

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

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

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
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

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 Pair-wise 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

#	Item	Pred. (Group)	Pred. (User 71)	Pred. (User 390)	Pred. (User 467)	Disagreement
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
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.

How to use the application

Implementation details and instructions on how to run the application and use it are specified in the README of the Github project.

References

[1] Amer-Yahia et al: Group Recommendation: Semantics and Efficiency (https://dl.acm.org/)