Recommender Systems

DATA.ML. 360-2020-2021-1

Sequential Recommendations

Kostas Stefanidis
konstantinos.stefanidis@tuni.fi

Fall 2020

https://moodle.tuni.fi/course/view.php?id=10515&lang=en

Recommenders

Recommendations have been integrated in many of the services available to users in recent times

- They provide the users with a list of items that are most relevant to them

Recommenders are employed to make the user experience better and smoother: listening to music, health information, queries, jobs

<u>The content-based method</u>

- The system recommends to the user items that are similar to other items that the user has already consumed in the past - requires knowledge about the items which is often not available

<u>The collaborative filtering method</u>

- Given a target user, find similar users to him/her, predict the relevant items for the input user based on the similar users preferences/ratings

Group recommendations

Especially due to the expansion of social media: group recommendations

- Instead of a single user requesting recommendations, it is more likely that a group will make a query

E.g.: Friends want to see a movie / Each friend has his/her own likes and dislikes / The system needs to balance them and offer to the group items that are somehow relevant to all member

Two main ways for doing so:

- Create a pseudo-user by combing the preferences of each group member, and apply a standard recommendation method
- Apply a recommendation method to each member individually, and aggregate the separate lists into one for the group

Aggregations in recommenders

Fairness: A important criterion that one can take into account during the aggregation stage

Recommend the item that has the best relevance to the group

How (Average): All group members are considered equals - Calculate the average score across all the group members preference score for an item

- *Drawback*: Assume a group of 3 - 2 of them are quite similar to each other, while the 3rd is not. By using the average method the opinion of the last user is lost

How (Minimum): The user with the minimum preference score will act as veto to the rest of the group

- *Drawback*: Consider the opinion of one group member - "ignore" the others
    - The system in most cases will recommend an item that is acceptable for all members, but it will be an item with a somehow low preference score

The need to combine the equality of the average method and the inclusivity of the least misery method is the driving force behind our work in this paper

Multiple iterations

But something is hidden in the picture!

We imply that each time a group is using the system is distinct from the previous ones - this is not a realistic scenario!

A group interacts with the system multiple times / The system should recommend different items at each iteration & retain knowledge from past interactions to keep the output diverse

- We want a system that has a memory and can adjust its recommendations accordingly

Both the average and least misery approaches fail

- Average method: the out-flier user is never satisfied
- Least misery method: the system recommends movies not highly interested by anyone in the group

Goal

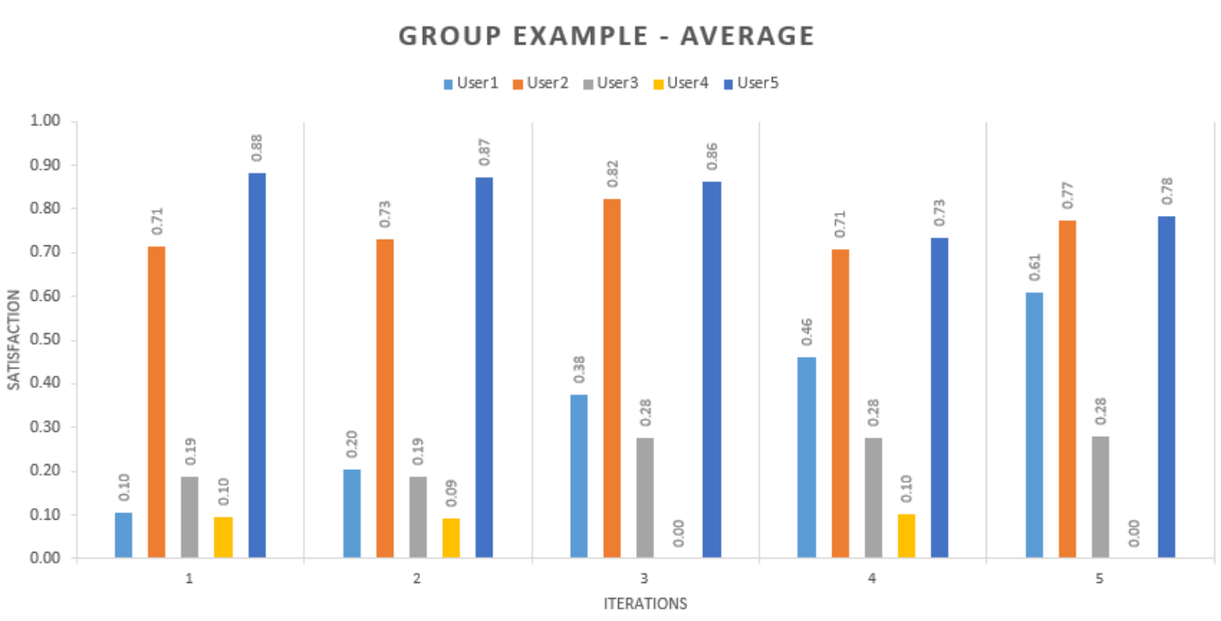
Goal: address the problem of unfairness generated by these methods when applied to sequential group recommendations

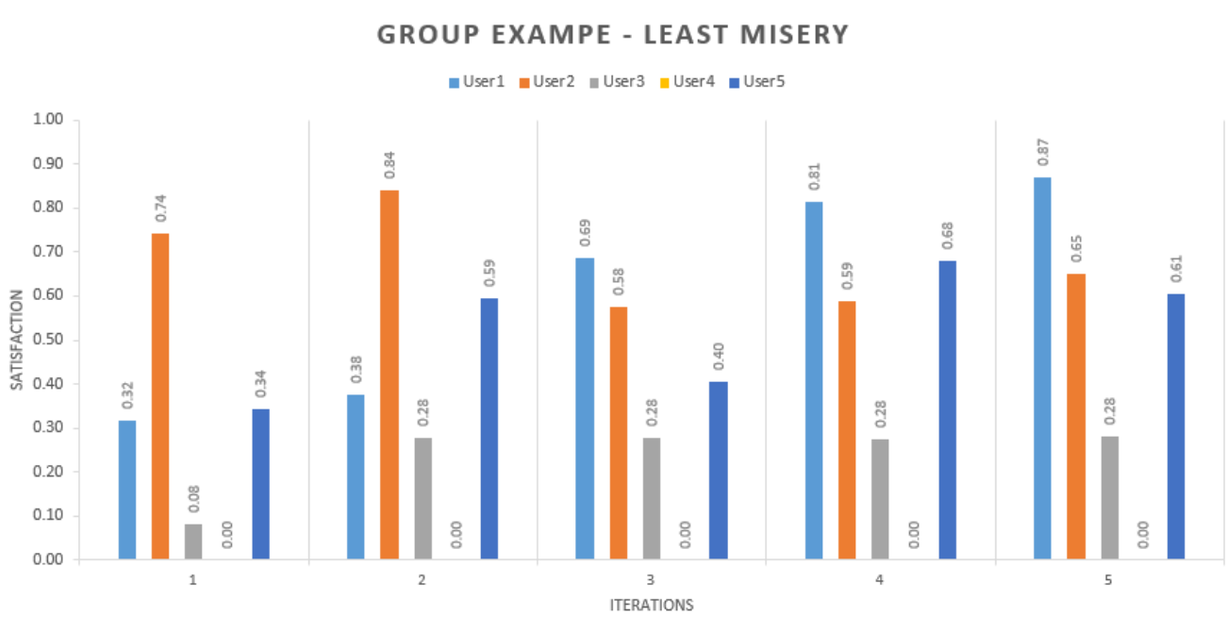
E.g.:

5 friends meet in regular times to watch a movie

Recommend the “best” 10 “new” movies for the group

Consider 5 iterations





Count the degree of satisfaction for each member

- Measure how relevant are the group list's items, over the best items for each group member

In both cases, **User 4** has a very low satisfaction score (for least misery is always 0), which implies that almost none of the reported items are of interest to him/her

The recommender is unfair to him/her - unfairness continues throughout the 5 iterations

Ideally: each new group recommendation should take into account what happened in the past. Not only what movies have already been recommended, but at which degree each member of the group was satisfied with that recommendation

Goal

Exploit here the idea of multiple iterations of recommendations: If a user is not satisfied in a round, then he/she was either satisfied in a previous or will be satisfied in the next round

Satisfaction to estimate the happiness of each group member after each iteration of the system

- Apply satisfaction scores during the aggregation phase as weights on the individual preference scores of the group members
  - The users not satisfied in the previous round will have more weight in the next
  - A member is not continuously biased against (as in *average*), since the calculation of the satisfaction scores is done dynamically at each iteration
  - We take into account the opinions of the entire group (something that *least misery* was lacking)

Basic concepts

Typical recommender without considering any past interactions: Aggregate individual group members preference lists into one group preference list

U: set of users / I: set of items / Ratings from 1 to 5: $r(u_i,d_z)$

$U(d_z)$: users that rated an item d_z / $I(u_i)$: items rated by a user u_i

G: Group of n users

- Apply a single user recommendation algorithm to all users in G and produce their suggestions $Au_1, \ldots, Au_n$

- Aggregate these lists into one

- Each item will have just one group preference score and the top k will be reported back to the group

Basic concepts

Typically, two well established aggregation methods are used

Average: All members are considered equals. So, the group preference of an item will be given be averaging its scores across all group members

$$avgG(d_z, G) = \frac{sum_{u_i \in G} p(u_i, d_z)}{|G|}$$

where p(ui,dz) gives us the preference score of dz for user ui

Least misery: One member can act as veto to the rest members of the group. In this case, the group preference score of an item dz is the minimum score assigned to that item in all group members preference lists

$$minG(d_z, G) = min_{u_i \in G} p(u_i, d_z)$$

Sequential Group Recommendations

We do not consider each group query to the system as a stand alone process, but as a sequence of queries submitted to the system by the same group

I.e., multiple iterations: each iteration has the main components of a standard group recommendation method, that is, single user recommendations followed by the group aggregation

GR: A sequence of μ group recommendations (Gr1,…,Grμ)

- pj(ui,dz): preference score of user ui for item dz at iteration j
- gpj(G,dz): preference score of item dz for the group G as a whole, as estimated by the group recommender at iteration j

Sequence of recommendations: If a user was not satisfied in a previous iteration, potentially, he/she will be satisfied during the current one

Single user satisfaction

To examine the effectiveness of our group recommender through a series of iterations, we introduce the notion of satisfaction per iteration

Our goal is to directly compare the user's satisfaction from the group recommendation list with the ideal case for that user

$$sat(u_i, Gr_j) = \frac{GroupListSat(u_i, Gr_j)}{UserListSat(u_i, A_{u_i,j})}$$

$$GroupListSat(u_i, Gr_j) = \sum_{d_z \in Gr_j} p_j(u_i, d_z)$$

$$UserListSat(u_i, A_{u_i,j}) = \sum_{d_z \in A_{u_i,j}} p_j(u_i, d_z)$$

Note: We use the scores as they appear in the individual preference list of the user - aggregations may distort individual opinions of the group members

Still static!

Overall user satisfaction

Overall satisfaction over μ iterations

$$satO(u_i, \mathcal{GR}) = \frac{\sum_{j=1}^{\mu} sat(u_i, Gr_j)}{\mu}$$

Group satisfaction

The satisfaction of the group G with respect to a group recommendation list Grj is defined as the average of the satisfaction of the users in the group:

$$groupSat(G, Gr_j) = \frac{\sum_{u_i \in G} sat(u_i, Gr_j)}{|G|}$$

The overall group satisfaction of G for a recommendation sequence GR is:

$$groupSatO(G, \mathcal{GR}) = \frac{\sum_{u_i \in G} satO(u_i, \mathcal{GR})}{|G|}$$

This measure indicates if the items we report to the group, are acceptable to its members

- *However, due to average, the dissatisfaction of a user can be lost in the computations*

Disagreements

Check for potential disagreements between the users in the group

$$groupDis(G, \mathcal{GR}) =$$
$$max_{u_i \in G} satO(u_i, \mathcal{GR}) - min_{u_i \in G} satO(u_i, \mathcal{GR})$$

Intuitively, we define the disagreement of the group, to be the difference in the overall satisfaction scores between the most satisfied and the least satisfied member in the group

We want this measure to take low values: this will indicate that the group members are all satisfied to the same degree

- Higher values mean that at least one member of the group is biased against

Problem definition

The Fair Sequential Group Recommendation Problem

For a group G, the sequential group recommender produces at the μ iteration a list of k items Gr_μ , $Gr_\mu \in GR$, that:

(1) Maximizes the overall group satisfaction, groupSatO(G,GR), and

(2) Minimizes the variance between users satisfaction scores, groupDis(G,GR)

Which is the role of the similarity between the group members?

Assume a group with two highly similar users and one very dissimilar to them

- High groupSatO, for increasing average satisfaction, needs to recommend items relevant to the similar users
- By doing so, groupDis takes high values as well, since we do not address the needs of the third user

Overall Recommend items not just good enough for all members – since that still returns low groupSatO values, *but items highly relevant to the group, without sacrificing the opinions of the minority*

Sequential hybrid aggregation model

For sequential recommendations, both the average and the least misery aggregation methods have drawbacks and advantages

- E.g., equality for average, and inclusion of all opinions for least misery

Sequential hybrid aggregation method: A weighted combination of them

$$score(G, d_z, j) =$$
$$(1 - \alpha_j) * avgScore(G, d_z, j) + \alpha_j * leastScore(G, d_z, j)$$

When $\alpha = 0$, we satisfy (1) : the average aggregation considers the best options for the group as a whole

When $\alpha = 1$, we satisfy (2) : if $\mu \geq |G|$ then each member has been satisfied at least once

**However, the benefits of $\alpha = 1$ can been seen for higher values of μ, since at the first iterations the group disagreement may remain high, since it considers the overall satisfaction of the users

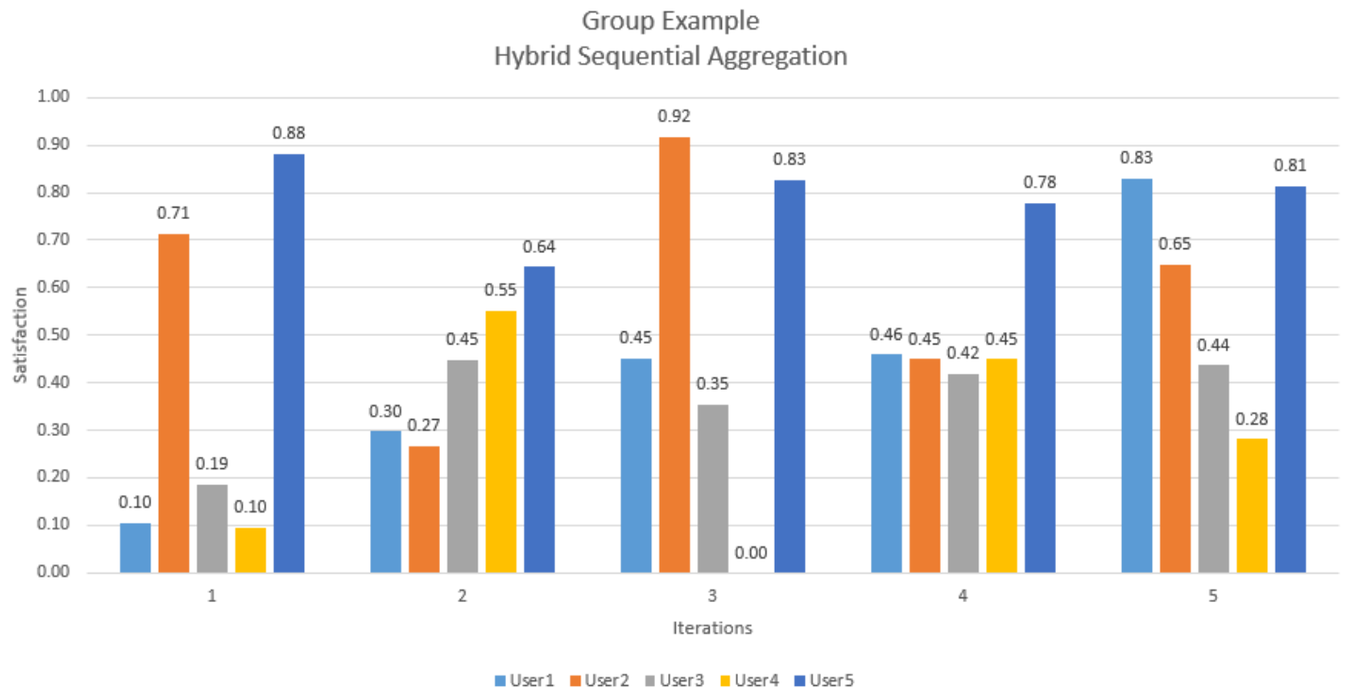
Sequential hybrid aggregation model

To satisfy both conditions, we need to set the value of α between 0 and 1

We set the value of α dynamically in each iteration by subtracting the minimum satisfaction score of the group members in the previous iteration from the maximum score (for j = 1, α = 0):

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}) - min_{u \in G} sat(u, Gr_{j-1})$$

- If one group member is extremely unsatisfied in a previous iteration, then α takes a high value and promotes that member's preferences
- If the group was equally satisfied at the last round then α takes low values, and the aggregation will closely follow that of an average, where everyone is treated as an equal



Sequential hybrid aggregation method

A group member that was not satisfied in the previous iteration of the system, is satisfied in the next

- Consider User 4: In the first iteration has a very low satisfaction score, and in the second has a higher one
  - Clear improvement over the previous results, where User 4 was always the least satisfied member of the group

Experimental setup

20M MovieLens Dataset: 20.000.263 ratings, 27.278 movies, 138.493 users

Sort the ratings based on time they were given / Divide the dataset into two parts of roughly the same size

The 1st part will be the starting dataset of our recommender: 8.381.255 ratings, 73.519 users, 6.382 movies

Split the remaining into chunks based on the timestamps

- Firstly, the system has access to the initial dataset
- When that iteration ends, the system is enhanced with the next chunk, and so on

The # of movies in each chunk increases as the time passes, since the users can rate old movies as well

The # of users is relatively the same across all chunks

Year	Semester	#Ratings	#Users	#Movies	#New Users	#New Movies
2004	1st	609499	5795	7395	5795	7395
	2nd	560550	5466	7589	2934	636
2005	1st	1157767	9021	8034	6191	450
	2nd	645391	6649	7991	3092	288
2006	1st	616616	6681	8196	3220	204
	2nd	555220	6508	8306	2996	234
2007	1st	585850	6493	8872	3020	288
	2nd	467580	5730	8761	2398	168
2008	1st	496838	6296	8613	2795	268
	2nd	661939	7430	9525	3998	739
2009	1st	521775	6841	10308	3136	1035
	2nd	408261	5907	10983	2432	1084
2010	1st	415410	6076	11657	2444	1192
	2nd	488281	6644	12280	3123	1206
2011	1st	435553	6560	12585	2944	1166
	2nd	330813	5693	12676	2181	1135
2012	1st	385523	5866	12716	2510	919
	2nd	345866	5433	12412	2206	1132
2013	1st	322737	5223	13034	2052	1157
	2nd	276590	5134	12575	2272	1302
2014	1st	252293	4710	12983	1913	1256
	2nd	310595	4612	13775	1917	1613

Group formation

Stable groups: the members of the group do not change between iterations

3 different types of groups, based on the similarity shared between the members (Pearson Correlation: values in [-1, 1])

$$s(u_i, u_l) = \frac{\sum_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})(r(u_l, d_z) - \bar{r}_{u_l})}{\sqrt{\sum_{d_z \in X} (r(u_i, d_z) - \bar{r}_{u_i})^2} \sqrt{\sum_{d_z \in X} (r(u_l, d_z) - \bar{r}_{u_l})^2}}$$

2 users are similar, if sim>0.5 / 2 users are dissimilar, if sim<-0.5

Groups types:

- 4 similar – 1 dissimilar
- 3 similar – 2 similar
- 3 similar – 1 dissimilar – 1 dissimilar

Experiments

For each group type, 100 different groups

For each group, perform sequential group recommendations

Calculate the average of groupSatO(G, GR) and groupDis(G, GR)

Recommend items the group has not previously seen: top 20 items

For predictions (use the top 100 most similar users to a given user):

$$p(u_i, d_z) = \bar{r}_{u_i} + \frac{\sum_{u_l \in (P_{u_i} \cap U(d_z))} s(u_i, u_l)(r(u_l, d_z) - \bar{r}_{u_l})}{\sum_{u_l \in (P_{u_i} \cap U(d_z))} |s(u_i, u_l)|}$$

$\alpha = 0$ - average aggregation method

$\alpha = 1$ - least misery aggregation method

Experiments

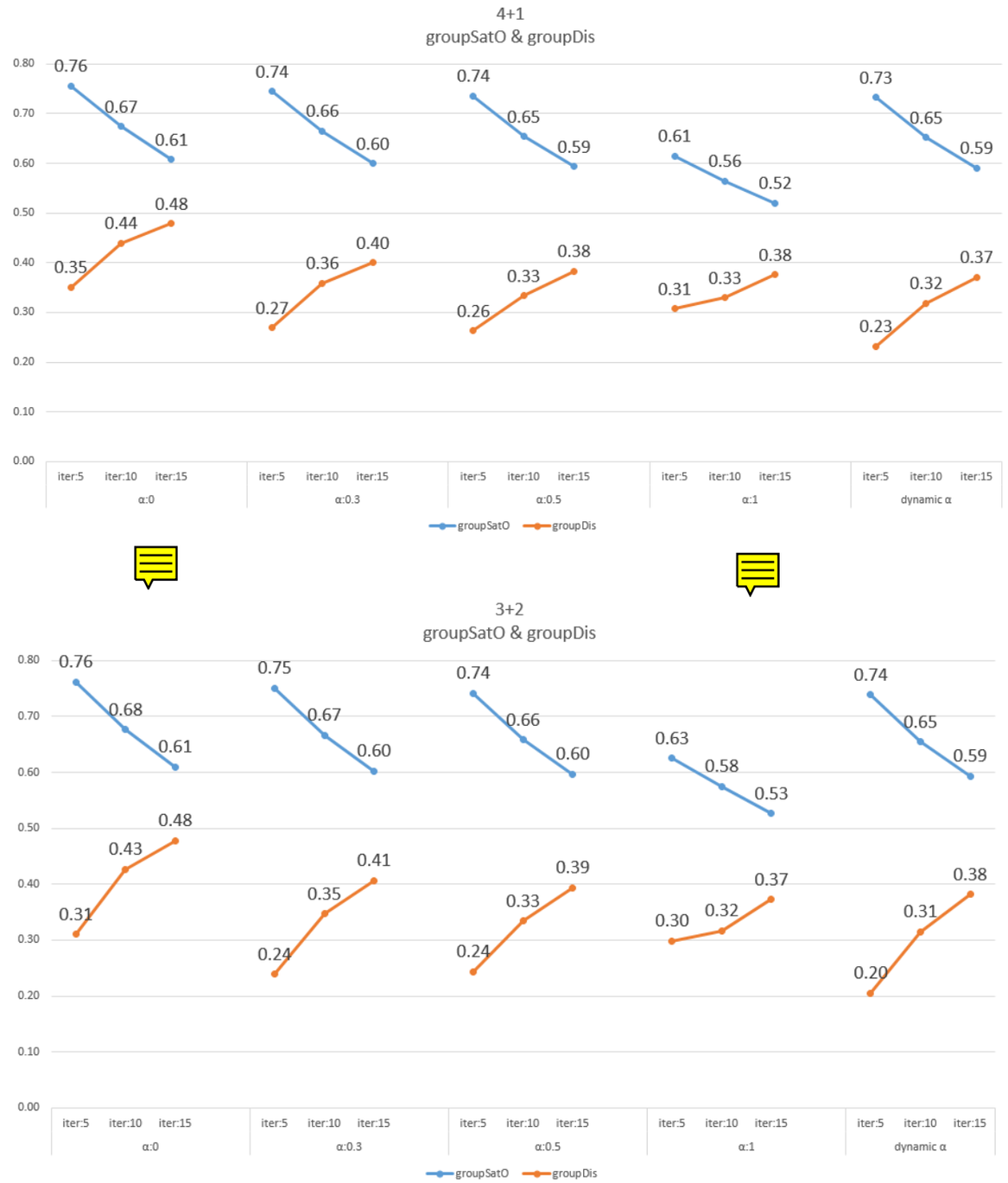
Overall group satisfaction: for all group types, decreases with the number of iterations

- In the latter iterations, the best items for the group have already been reported

Group satisfaction for different α values: the average method slightly outperforms the dynamic α

- The average method tries to offer the best results to the group as a whole

By far the worst performance is for α = 1, since in each iteration we consider only one user



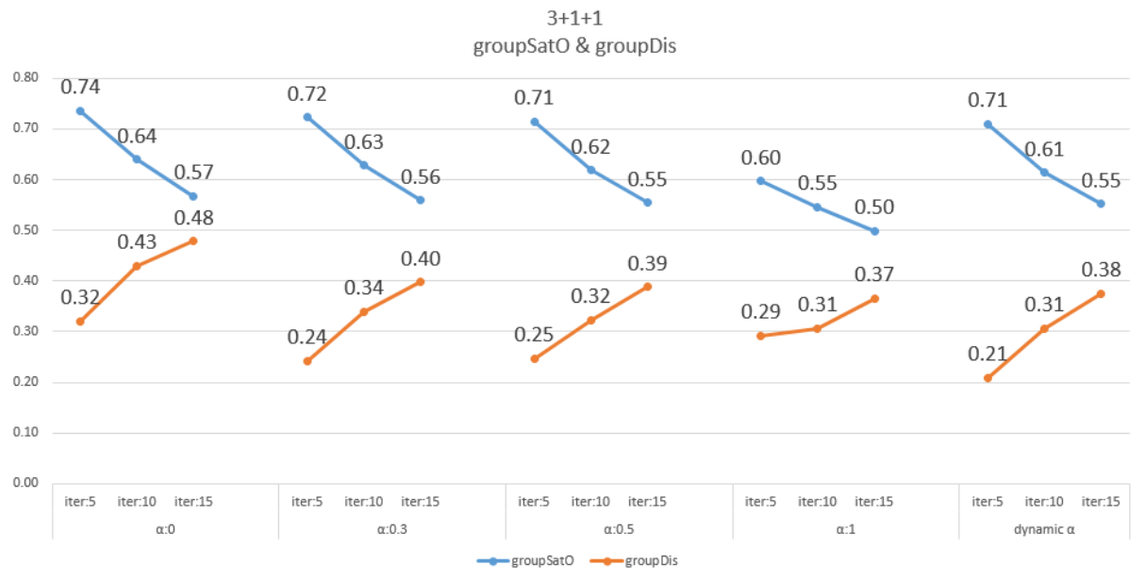
Experiments

Group disagreement

$\alpha = 0$ has the worst results / $\alpha = 1$ and dynamic α have the best results

- Average ignores the minority opinion in the group, which is reflected in the high group disagreement values
- The least misery method gives greater value on the opinions of the minority, leading to low disagreement values

Dynamic approach: small loss of group satisfaction, having the advantage of small disagreements



Future work

Ephemeral groups

Thanks

Experiments

100 groups per group type, per iteration (at the first iteration the α takes the default value 0)

During the next iterations the α values increase in a close to linear form

- This is expected, since by design the group formation have at least one out-flier member that is dissimilar to the rest of the group members
- This guarantees that at least one member is not satisfied during an iteration of the system

The high values of α in the later iterations are also the result of the dataset splitting

- At the later iterations, most of the dataset has already been given as input to the system, and statistically, the best items for the group and the individual members have already been recommended