GROUP RECOMMENDATIONS: SEMANTICS AND EFFICIENCY

Due to the growing of the people's activities on the Web, the need of finding relevant content on there for the users has grown as well.
The problem of finding relevant content for a group its named group recommendation problem.

The first difficulty that we found in the group recommendation problem is the fact of set the semantics of that group. In a individual form is quite easy due to the fact that the usual method is getting the elements with the higher score or rating value. But in a group recommendation the value of a element may be different for each component of that group.
A few methods have focused on aggregating individual group members' relevances to produce recommendations to a group. They can be classified in two groups: *Preference aggregation* and *score aggregation.* Between both, score aggregation is a better approaching, it offers better flexibility and is more efficiency.
The second difficulty is how to compute group recommendations given a consensus function on a efficient way. On the given document, we are going to analyze and study how to approach this type of recommendations.

Model and semantics
For the next explanations we are going to use the *U* as the set of users(u) and *I* as the set of items(*i*). We can suppose that each user has already provided a rating or a value for each item between the 0-5 values, this is denoted as **rating(***u,i***)**. There is another function called **relevance(***u,i***)** which calculates the relevance scores for each pair of users-items.

Starting on the **Individual Recommendation** the most common recommendation strategies are based on finding items that are similar to the higher rated item's user(item-based), or finding users that actually share the same interests(collaborative filtering).
In the item-based cases, the relevance value is calculated with:

$$relevance(u, i) \ = \sum_{i' \varepsilon I} ItemSim(i, i') \ \times rating(u, i')$$

Where *ItemSim(i,i')* calculates the value of similarity between the item *i* and *i"*.

In the collaborative filtering, the relevance is calculated in the next way:

$$relevance(u, i) \ = \sum_{u' \varepsilon U} UserSim(u, u') \ \times rating(u', i)$$

Where *UserSim(u,u')* returns the value of the connectivity between both users *u and u'.*

Approaching the **Group Recommendation**, we have the main difficulty which is that each user has a different rating value for the same item. Due to that, it's pretty difficult to design a *consensus score* for each item. The two main aspects to the *consensus score* are the *group* *relevance (reflect the degree to which the item is preferred by the members)* and the *group disagreement (reflects the level at which members disagree which each other)*

The two aggregation strategies considered for approaching the **Group Relevance *(rel(G,i))*** value are:

-Average: $rel(G,i) = \frac{1}{|G|}\Sigma(relevance(u,i))$

-Least-Misery : $rel(G,i) = Min(relevance(u,i))$

For measures the **Group Disagreement *(dis(G,i))*** we consider two disagreement computation methods:

-Average Pair-wise Disagreements:

$dis(G,i) = \frac{2}{|G|(|G|-1)}\Sigma(|relevance(u,i) - relevance(v,i)|)\ where\ u \neq v\ and\ u,v\ \varepsilon\ G$

-Disagreement Variance:

$dis(G,i) = \frac{1}{|G|}\sum_{u\varepsilon G}(relevance(u,i) - mean)^2$ *where **mean** is the the mean of all the individual relevances for the item.*

Combined both Group Relevance and Group Disagreement we can get the **Consensus Function *(F(G,i))*.** Its formula:

$F(G,i) = w_1 \times rel(G,i) + w_2 \times (1 - dis(G,i))$ *where* $w_1 + w_2 = 1.0$ *and each specifies the relative importance of relevance and disagreement in the overall recommendation score.*

We adopt this consensus function because of its simplicity and the fact that can be easily applied for the computation.
Now that we have the necessary function, we can approach the problem (**Top-k Group Recommendation**). Given a user group G and a consensus function F, identify a list I_G of items that contains the recommended items for that group sorted on decreasing group recommendation value.

Applicability of Top-k threshold algorithms
Threshold algorithms can be used for computing the Top-k items problem. This is possible thanks to the monotony of the consensus function. It is clear that the group relevance functions proposed before are mononote as well, but it’s unclear for the group disagreement functions. One of our purposes then is try to demonstrate that they are also monotonic.
For prove the **Monotonicity of Group Disagreements** we are going to maintain *pairwise disagreement lists.* A pairwise disagreement list for users *u* and *v* is a list of items which are sorted in the increasing order of the difference between their relevance scores for *u and* v.
Using this lists is enough to compute disagreement variance in a monotonic fashion. Still, materializing all the possible pair-wise disagreement it’s not practical due to the fact that the number of them grows in a quadratic way due to the number of users.

Efficient Computation of Group Recommendation
Back in to the main problem of recommendations, there are a few **Group Recommendation Algorithms** that we are going to analize, starting with:

-Group Recommendation Algorithm with Fully Materialized Disagreement Lists(FM):
This one uses relevance lists IL of each user in the input group G and disagreement lists DL for every pair of users in G. Each relevance list is obtained with one of the individual recommendation strategy, and each disagreement list is generated for a user pair and saves the difference in scores for all items in their respective relevance lists.
This algorithm makes sequential access on each input lists in a round-robin fashion. It uses two routines, one for compute the score of the current item. Its performance is based on random access on all the relevance lists.
The other routine is for produce a new threshold value at each round. Its main purpose is to provide an upper bound the score of any item that has not yet been seen by the algorithm. This is computed as:

$$F(G,i) \leq w_1 \times \frac{1}{|G|} \sum_{u \varepsilon G} r_u + w_2 \times (1 - \frac{2}{|G|(|G|-1)} \sum_{u,v\ \varepsilon G} \Delta_{u,v})$$

-Group Recommendation Algorithm for Relevance lists Only(RO):
This is a variation of the previous algorithm which applies only when the relevance lists are present and none of the disagreement lists are available. His main benefit is the fact of using less space. The lack of a disagreement list doesn't affect the first routine of the previous algorithm, but it does affect the second routine, so it has to be modified. Since they are not available, we assume the the pairwise disagreement between each pair of users for any unseen item is 0. Then, the upper bound for the threshold value is calculated as:

$$F(G,i) \leq w_1 \times \frac{1}{|G|} \sum_{u \varepsilon G} r_u$$

-Group Recommendation Algorithm for Partial Materialization(PM):
This variant is used where only some disagreement list are materialized. As the RO, this variant has the same benefit of using less space. The main differences with the others is how the threshold is computed:

$$F(G,i) \leq w_1 \times \frac{1}{|G|} \sum_{u \varepsilon G} r_u + w_2 \times (1 - \frac{2}{|G|(|G|-1)} \sum_{(u,v) \varepsilon M} \Delta_{u,v})$$ *where M is the set of all pairs of users that has their disagreement lists materialized*

Based on this idea, anyone may think that the more disagreement lists are materialized, the tighter the score bound and hence, the faster the algorithm terminates. But it turns out that it's not always like that. Then which pair-wises disagreement list should be materialized?

For the **Partial Materialization** it is more practical to materialize only a small but effective subset of the disagreement lists in a pre-processing step. But how we determine which lists

have the ‘max. Benefit’ to materialized them? They should be materialized if the corresponding users together are very likely to be a part of user groups seeking item recommendations or if materializing the list significantly improves the running time of top-k recommendation algorithm. The problem then is to determine the partial materialization that comply both requisites. This problem is actually quite hard to solve in a optimal way. Next there are a several principles and practical solutions to that problem.
On the first approach, we assume that each future user group query *G* will only contain two users and the probability of G being the next ‘query’ (*p(G)*) is reliably known for all pair of users G. Then, after compute M(subset of user pairs whose corresponding disagreement list we would like to materialize), we have to show that the materialized disagreement lists can be used at query time for answering any user group *G,* even in groups with more than two users.
A disagreement list is useful for forcing early termination only there is significant skew in its disagreement scores.
In a large user base it’s critical that we identify only those user pair that had significant likelihood of occurring together. We have to know that we’re computing the probabilities for the user pairs that occur in the query log, so we can avoid examining all the rest user pairs that never occur together.

We can modify the behaviour of the second routine’s algorithm(produce a new threshold value at each round) making him to compute sharper thresholds in each iteration, so we can improve the algorithm’s performance drastically. This is possible due to the dependencies between both lists (relevance and disagreement). We are faced with a formal optimization problem after each iteration where we look for maximize the consensus function over |*G*| real-valued variables. This optimization problems have really complex formulations, but in the other hand, the size of the problems themselves are very small, consisting of only a few variables and constraints. They can be solved by general purpose non-linear solvers with no overhead per iteration.

Experiments
The experiments are going to analyze the group recommendation system in two different perspectives..
First, we are going to start with the *quality* perspective and after that, the next and the last perspective is going to be the *performance* one.

For the *quality* perspective we compare four group recommendation mechanisms:
-Average Relevance Only (AR):
Computes an item score as its average relevance among group members. The disagreement weight is set to 0.
-Least-Misery Relevance Only(MO):
Computes an item score as its minimum relevance among all group members. The disagreement weight is set to zero.
-Consensus with Pair-wise Disagreement (RP):
Computes an item score as a weighted summation of its average relevance and its average pair-wise disagreements between all group members.

Consensus with Disagreement Variance (RV):
Computes an item score as a weighted summation of its average relevance and the variance of its relevance between all group members.
The **User Study** is separated in two phases *User collection Phase* and *Group Judgment Phase.*
On the first phase, we divide the movies in two sets, one is the popular set and the other is the diversity set. There are 3 groups, *similar, dissimilar* and *random.* The first one is formed with users who have completed the Similar HIT(Human Intelligence Tasks with the popular set) combined with having the maximum summation of pair-wise similarities among all groups of the same size.The second is formed with the users that have completed the Dissimilar HIT combined with having the minimum summation of pair-wise similarities among all groups of the same size. The last group is formed randomly selecting users from all the available of them.

In the second phase (*Group Judgment Phase*) the objective is to obtain ground truth judgments that we are going to use to compare group recommendation generated by the four mechanisms named before.
On the **individual recommendation** we generated and materialized a list of individual recommendations using collaborative filtering.
On the **Group Recommendation Candidates** for each group we generated group recommendations using all the strategies.
Given a Mechanical Turk user's ground truth evaluation of the candidate movies, we evaluate each of the following six group recommendation strategies:
AR, MO: Group recommendation lists generated based on average relevance and least-misery, respectively.
RP20, RP80:Group recommendation lists generated by combining group relevance (average relevance) and pair-wise disagreements.
RV20, RV80:Group recommendation lists generated by combining group relevance (average relevance) and disagreement variance.

Each strategy generates a 10-movie recommendation list and for a given list, its Discounted Cumulative Gain value is calculated as follows:

$$DCG_{10} = rel_1 + \sum_{i=2}^{10} \frac{rel_2}{log_2(i)}$$

Where rel_i is the ground truth of the movie at position i.
Based on the results given by using this strategies, we conclude that for similar users, MO results have the best performance for small and big groups. The next best strategy is AR. RV80 and RP80 had the worst performance due to the fact that there is hardly scope of difference on opinion in a group of similar users.
But when we talk about dissimilar users we can realise that RV80, RP80 start to perform better than the other two strategies. Specifically for large groups, RV80 gives the best results of group recommendation. Last, for the random users, MO works really well when it's about small groups, but when it's about large groups, all the strategies had the same performance

Now, experimenting the **Performance Evaluation** we are going to use the three algorithms: Dynamic Computation with Relevance List Only (RO), Full Materialization (FM), and Partial Materialization with a given budget on number of lists (PM).
For comparing this algorithms we are just going to review the number of sequential accesses.
The **Group Formation** was realized by selecting users from the MovieLens database. The key factor considered to pick them was the similarity. Other factors were the number of recommendations being produced(K) and the size of groups.
When analysing the results we realised that the group similarity has a direct impact on the number of sequential accesses(SA). The second observation that we saw is that some Disagreement lists almost always guarantee earlier stopping, but this doesn’t happens always. For different user groups different strategies will win.
If we vary the group similarity and we increase the group, the effectiveness of our materialization algorithms decreases. This happens because the more similar the members are with each other, the more likely their agreements on the top items are close to the upper bounds that are estimated in the RO algorithms. As a result, RO can reach stopping conditions as Earlier than the other two algorithms can reach them. In conclusion we can say that for very high similarities, RO gives the best results on performance, but for very low similarity cases, FM is the winner in most of the cases. The performance of the PM is between both RO and FM performance, we can say then that has an average performance.
If we vary K (number of items recommended), the number of sequential accesses increases as the same time we increase the value of K. In this cases of varying K, algorithm PM has a way better performance than the others.
If we vary the group size, we can observe that the sequential accesses increases at the same time as the group size does. When the group sizes are small and medium, both materialization algorithms act significantly better than RO.

What is the impact of materializing different numbers of disagreement lists?
We can see that when the number of DL is 0, the performance is really bad, but the more DL we add, the performance gets better. However, there is a point when the performance starts to degrade and never gets better again

With the last analysis of the effect of threshold sharpening we can observe that FM with optimization is never worse than FM without optimization. In fact, it is easy to observe that the effect of optimization becomes significant with higher values of k.

G.Fernando Lojano Mayaguari & Miguel Moraga Muñoz