Preference Elicitation for Narrowing the Recommended List for Groups
Lihi Naamani-Dery\textsuperscript{1,3}, Meir Kalech\textsuperscript{2}, Lior Rokach\textsuperscript{2,3}, Bracha Shapira\textsuperscript{2,3}
\textsuperscript{1}Department of Industrial Engineering and Management at Ariel University, \textsuperscript{2}Department of Information Systems Engineering at Ben-Gurion University, \textsuperscript{3}Telekom Innovation labs at Ben-Gurion University
\{ln,kalech,liorrk,bshapira\}@bgu.ac.il

\section*{ABSTRACT}
A group may appreciate recommendations on items that fit their joint preferences. When the members’ actual preferences are unknown, a recommendation can be made with the aid of collaborative filtering methods. We offer to narrow down the recommended list of items by eliciting the users' actual preferences. Our final goal is to output top-\(k\) preferred items to the group out of the top-\(N\) recommendations provided by the recommender system (\(k < N\)), where one of the items is a necessary winner. We propose an iterative preference elicitation method, where users are required to provide item ratings per request. We suggest a heuristic that attempts to minimize the preference elicitation effort under two aggregation strategies. We evaluate our methods on real-world Netflix data as well as on simulated data which allows us to study different cases. We show that preference elicitation effort can be cut in up to 90\% while preserving the most preferred items in the narrowed list.

\section*{Categories and Subject Descriptors}
H.3.3 \[\{\text{Information Storage and Retrieval}\}: \text{Information Search and Retrieval}\]

\section*{General Terms}
Algorithms, Human Factors, Experimentation

\section*{Keywords}
Preference Elicitation, Group Recommender Systems

\section{1. INTRODUCTION}
Group recommendation may be useful in various common daily scenarios. For example, a group who wishes to dine together may appreciate recommendations on relevant restaurants that fit their joint preferences [11]. Given the list of recommended restaurants, the group’s members often interact among themselves to narrow down the possible options and choose a restaurant [10]. When the members’ actual preferences regarding the recommended items are known, some aggregation strategy can be used to compute the most preferred restaurant out of the list of recommended items. However, often full preferences are not readily available, mainly because it can be tiresome or difficult for users to determine all of their preferences in advance. Consider for example a setting of 30 optional dinner locations for a group. The group members might be reluctant to provide input on each of the restaurants.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

The definitive version was published in Proceedings of the 8th ACM Conference on Recommender Systems, October 2014. Og 333-336. \url{http://doi.acm.org/10.1145/2645710.2645760}

Copyright 2014 ACM 978-1-4503-2688-1/14/10...\$15.00.
aggregation strategy affects the effort required in the preference elicitation and evaluate two state-of-the-art aggregation strategies.

Our final goal is to output top-k preferred items to the group out of the top-N recommendations provided by the recommender system (kNS), where one of the items is a necessary winner. We propose an iterative preference elicitation method, where users are required to provide item ratings per request. We suggest a heuristic that attempts to minimize the preference elicitation effort under two aggregation strategies, while still outputting a necessary winner. We evaluate our methods on real-world NetflixB. In the following, we assume that users’ preferences are unknown in advance, but can be acquired during the process, i.e., if a user is asked for her preference on an item she is able to submit it. We also assume a user submits her true preferences and thus in this paper we do not consider manipulation. Lastly, we assume an approximate distribution of each user’s preferences for each item exists, as in [10].

2. RELATED WORK
Both [9] and [13] study how different strategies affect group members. However, the aggregation strategies have not been studied in the context of preference elicitation. Practical preference elicitation is gaining interest recently. In [6] it is assumed that each user holds a predefined decreasing order of the preferences. In an iterative process, the users are requested to submit their highest preferences; the request is for the rating of one item from all the users. However, requiring the users to predefine their preferences can be inconvenient to the users. In [10] two practical heuristics that attempt to minimize preference elicitation effort are presented for the Range voting protocol. However both output one definite item and not top-k items. In [8] a practical elicitation process is proposed for the Borda voting protocol. The output is multiple winners. However the authors did not consider the Range protocol. In all the above, the authors considered only the majority based aggregation strategy. The algorithm proposed in [8] lacks details and the authors did not provide us with explanations so we cannot expand it to the Least Misery strategy or to the Range voting protocol. In this paper we expand the preference elicitation algorithm for the Range voting protocol suggested in [10]. To the best of our knowledge the issue of preference elicitation and returning one or more items under the Least Misery strategy has not been investigated before.

3. MODEL DESCRIPTION
Let us define a set of users (voters) as \( V = \{ v_1, v_2, \ldots, v_n \} \) and a set of candidate items as \( C = \{ c_1, c_2, \ldots, c_m \} \). When queried, the users assign ratings to the items from a discrete domain of values \( D = (d_{\min}, \ldots, d_{\max}) \) where \( d_{\min} \) and \( d_{\max} \) are the lowest and highest values, respectively. User \( v_i \)'s preferences are represented by the rating function \( v_i : C \rightarrow D \). For abbreviation we denote \( q_i \) as a single preference of \( v_i \) for a single item \( c_j \). The user-item rating \( q_{ij} \) is not necessarily known to the voting center. Let \( O^i = \{ q_{i1}, \ldots, q_{in} \} \) represent the set of user \( v_i \)'s responses to queries. Note that this set does not necessarily contain all the items. \( O^i = \{ O^1, \ldots, O^n \} \) is a set of \( O^i \) sets. In an iterative process, in each round one user is queried for her ratings on one item. The preferences are aggregated according to a predefined aggregation strategy. The process terminates once a predefined termination condition is reached.

We denote the applied aggregation strategy \( \text{str} \). The item score depends on the strategy used. As mentioned, in the Majority based aggregation strategy the fundamental is towards the majority of the group. The user rating for an item is added: \( s_{\text{majority}}(c) = \sum_{i=1}^{n} q_{ij} \). Although the Majority Based Strategy is fairly common, a disadvantage of this strategy is that it can be unfair towards users with the minority view. In fact, the authors in [14] state that their system works well for a homogenous group. However, when the group is heterogeneous, dissatisfaction of the minority group occurs. In the Least Misery strategy, the chosen item cannot be the least preferred by any of the users. The items score is the score of the least preferred item: \( s_{\text{least}}(c) = \min_{i=1}^{n} q_{ij} \). An implementation can be found in [12] for example. The disadvantage is that the minority opinion can dictate the group – if all users but one really want some item to win, it will not be chosen [9].

Let \( k \) be the number of alternatives that will be presented to the group members. When given a set of responses to queries \( O^i \), the goal is to determine whether the iterative process can be terminated. The termination condition is that the winner item is definitely one of the \( k \) items. The function \( p_{\text{max}}(c_i, O^i) \) computes the possible maximum rating for item \( c_i \) given the known user preferences. Similarly, the function of the possible minimum rating is \( p_{\text{min}}(c_i, O^i) \).

Definition 1. (Possible Maximum): given the set of responses \( O^i \) and an aggregation strategy \( \text{str} \), the possible maximum score of candidate \( c_i \), denoted \( p_{\text{max}}(c_i, O^i, \text{str}) \), is computed as follows:

\[
p_{\text{max}}(c_i, O^i, \text{str}) = \begin{cases} \sum_{j \in 1..m} p_{\text{max}}(c_j, O^j) & \text{str} = \text{majority} \\ \min \left( \sum_{j \in 1..m} p_{\text{max}}(c_j, O^j) \right) & \text{str} = \text{least misery} \\ d_{\min} & \text{otherwise} \end{cases}
\]

Definition 2. (Possible Minimum): given the set of responses \( O^i \) and an aggregation strategy \( \text{str} \), the possible minimum score of candidate \( c_i \), denoted \( p_{\text{min}}(c_i, O^i, \text{str}) \), is computed as follows:

\[
p_{\text{min}}(c_i, O^i, \text{str}) = \begin{cases} \sum_{j \in 1..m} p_{\text{min}}(c_j, O^j) & \text{str} = \text{majority} \\ \min \left( \sum_{j \in 1..m} p_{\text{min}}(c_j, O^j) \right) & \text{str} = \text{least misery} \\ d_{\min} & \text{otherwise} \end{cases}
\]

A necessary winner \( \text{NW} \) is a set of items whose possible minimum aggregated rating is equal or greater than the possible maximum aggregated rating of all the others. Formally:

Definition 3. (Necessary Winners set): \( \text{NW} = \{ c_i | p_{\text{max}}(c_i, O^i) \geq \max_{c_j \in C} p_{\text{max}}(c_j, O^i) \} \) for \( c_i \in C \)

Although the necessary winner set may contain more than one item, we assume that there is only one necessary item. In the case of a few winning items, the first item is selected lexicographically. In order to decide on the next query, the elicitor considers the rating distribution of the user-item preferences. The distribution can be inferred from rankings of similar users using collaborative filtering methods [10]. Formally the rating distribution is:

---

Definition 4. (Rating Distribution): The voting center considers $q_j$ as a discrete random variable distributed according to some rating distribution $v_d$, such that $v_d[d_j] = P_r(q_j = d_j)$.

Where $v_d = [v_{d1}, v_{d2}, ..., v_{d_m}, v_{d_{m+1}}, ..., v_{d_{m+n}}]$ is the set of all the rating probability distributions.

The example presented in Table 1 shows the rating distribution of three users for two items in the domain $D = \{1,2,3\}$. E.g., the probability that $v_1$ will assign a rate of 1 to item $c_1$ is 0.2.

We assume independence between the probability distributions. While the independence assumption is naive, it can be used for approximating the actual probability. An attempt to address dependency will yield probabilities that are too complex for a system. When facing the tradeoff between the models accuracy and practicality, we chose to model a practical system. However, note that the precise probability value is not required if the queries are still sorted correctly according to the value of the information they hold (their informativeness). In the closely related domain of machine learning, a similar naive assumption is known to provide accurate classification, though the independence assumption is not always true [4]. We therefore argue that the decrease of accuracy, if at all exists, is not significant.

The initial rating distribution is calculated before the heuristics are applied. Both heuristics iteratively reveal one new rating upon request. This allows updating the distribution every time a new rating is added. The accuracy is expected to grow with the number of ratings acquired. Note that the proposed algorithm can be used when no history of ratings is available (this is known as cold start). In such a case, the initial distribution will be uniform, and will be updated as we proceed.

Table 1. Rating distribution of 3 users, $V = \{v_1, v_2, v_3\}$

<table>
<thead>
<tr>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>$c_3$</td>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>

4. The Preference Elicitation Heuristic

The Extended Dynamic Information Gain (EDIG) Heuristic is an iterative algorithm based on the DIG algorithm proposed in [10]. It uses a greedy calculation in order to select a query out of the possible $m \times n$ queries. The chosen query is the one that maximizes the information gain. The information gain of a specific query is the difference between the prior and the posterior probability of the candidates to win given the possible responses to the query. The algorithm terminates once a winner is within the requested top-$k$ items. In order to select a query, the heuristic calculates the information gained from each one of the optional queries and then selects the one that maximizes it. To compute the information gain, the winning probability of each item is calculated. The score $s$ that the candidate can achieve depends on the strategy:

$s = \{(n \cdot d_{\text{min}}, ..., s_{\text{majority}}, ..., n \cdot d_{\text{max}}) \text{ if } str = \text{majority} \}$

Let us denote the probability an item has a certain score $Pr(c_j = s)$.

The probability of an item to win is:

Definition 5. (Item Winning Probability): Under the independence of probabilities assumption, the probability that item $c_j$ is a winner is the aggregation of $c_j$’s probabilities to win over the possible ratings $s$: $Pr(NW = c_j) = \sum_{s \in \mathcal{S}} Pr(c_j = s) \cdot Pr(NW = c_j)$

To compute the probability that an item will receive the score $s$, and to compute the probability that an item will receive a score of at most $s$, for the majority based strategy we use:

$Pr(c_j = s) = \sum_{x=s}^{d_{\text{max}}} \prod_{i=1}^{n} Pr(q_i = s - x) \cdot Pr(q_{s,n} = x)$

For the Least Misery strategy we use:

$Pr(c_j < s) = \min_{x=s} \prod_{i=1}^{n} Pr(q_i = s - x) \cdot Pr(q_{s,n} = x)$

Next, the heuristic calculates the information gain of the $m \times n$ possible queries. The information gain of a query is the difference between the prior entropy and the posterior entropy, given the possible responses to the query:

$H(NW) = -\sum_{j=1}^{n} Pr(NW = c_j) \cdot \log(Pr(NW = c_j))$

Definition 6. (Information Gain): The Information Gain (IG) of a query is: $IG(NW|q_j) = H(NW) - \sum_{i=1}^{n} \min_{x=s} H(NW|q_i = d_x) \cdot Pr(q_i = d_x)$ where $H(NW|q_j) = d_x$ represents the entropy of NW given the possible values by querying user $v_i$ on item $c_j$.

The query that maximizes the information gain is selected: $\text{argmax}_{i,j} IG(NW|q_j)$. The query selection process continues until the termination condition is reached.

5. EVALUATION

We examine the performance of the DIG algorithm, under two aggregation strategies: Majority and Least Misery. The tradeoff between the number of items outputted (top-$k$) to the group and the number of preference elicitation required is investigated. Performance is measured in terms of elicitation effort, i.e., the required number of queries (requests for the user to provide ratings), in order to reach the termination condition. As mentioned in the related work, to the best of our knowledge there are no other algorithms that operate (or can be expanded to operate) under the same settings. Therefore the baseline for measuring the effectiveness of our method is a random procedure (RANDOM), which randomly selects the next query. To account for the randomness in RANDOM, each experiment was repeated 20 times.

We present an evaluation on simulated data as well as on a real-world dataset, the Netflix prize dataset. The Netflix dataset (http://www.netflixprize.com) consists of ~100,000 users and ~16,000 items. We consider a setting of a group of 10 members and 10 items. The users in the group are drawn randomly from a subset of Netflix where all users gave rating to 100 items. 90 items are used to create the initial rating distribution. The 10 remaining items are the items in question. To account for randomness, each experiment was repeated 10 times.

We simulate a cold start scenario where no history of ratings is available. We consider a setting of a group of 10 members and 10 items where the users rate their preferences on a scale of 1-4. We set the user-item distribution to a Uniform distribution. For example, the initial distribution for any user-item pair is: $(0.25, 0.25, 0.25, 0.25)$, indicating that item $c_j$ has a 25% probability to be rated 1-4 by user $v_i$. Similarly, we simulated a problem where one item is suspected to be more favorable and has a skewed distribution skewed to the right (Right-Skew), and a
problem where one item is suspected to be less favorable and has a distribution skewed to the left (Left-Skew). The distribution values for the right and left skews were set to: \{0.011,0.011,0.147,0.832\} and \{0.832,0.147,0.011,0.011\}. In all cases, we cast lots to set the user-item ratings. To account for randomness, each experiment was repeated 10 times.

We compared the two strategies: Majority (MAJ) and Least Misery (LM) combined with EDIG and RANDOM heuristics. We examined different termination conditions, from \(k = 1\), i.e., requiring one definite winner, to \(k = 9\), i.e., requiring the winner to be one of the top 9 items. The results on the Netflix data are displayed in Figure 1. Axis x presents the termination condition \(k=1,\ldots,10\). Axis y presents the number of queries required in order to find a winner within the top-\(k\). As expected, all combinations (of heuristics with strategies) drop in the amount of elicitation effort required as the termination condition includes a larger number of \(k\) items. The best results are received for EDIG combined with MAJ strategy. This combination reduces the elicitation effort in up to 35%. All the results were found significant with a 95% confidence level using a one-tailed t-test.

### Table 2. Preference Elicitation effort on simulated data

<table>
<thead>
<tr>
<th>Topk</th>
<th>Right-Skew</th>
<th>Uniform</th>
<th>Left-Skew</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAJ</td>
<td>LM</td>
<td>MAJ</td>
</tr>
<tr>
<td>1</td>
<td>27.7</td>
<td>83</td>
<td>79.7</td>
</tr>
<tr>
<td>2</td>
<td>26.1</td>
<td>67.8</td>
<td>70.1</td>
</tr>
<tr>
<td>3</td>
<td>24.7</td>
<td>54.5</td>
<td>72.7</td>
</tr>
<tr>
<td>4</td>
<td>23.2</td>
<td>44.1</td>
<td>70.1</td>
</tr>
<tr>
<td>5</td>
<td>21.2</td>
<td>35.9</td>
<td>65.2</td>
</tr>
<tr>
<td>6</td>
<td>19.3</td>
<td>28.9</td>
<td>59.7</td>
</tr>
<tr>
<td>7</td>
<td>17.4</td>
<td>23.9</td>
<td>55.5</td>
</tr>
<tr>
<td>8</td>
<td>15.2</td>
<td>17.3</td>
<td>50.3</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>14.7</td>
<td>35.1</td>
</tr>
</tbody>
</table>

For the simulated data with different skewness levels: Right-Skew, Uniform (cold-start problem) and Left-Skew, EDIG significantly outperformed RANDOM with a confidence level of 95% using a one-tailed t-test. For a cold-start problem (Uniform skewness, columns 4-5 in Table 2) LM outperforms MAJ reducing the preference elicitation effort in 22%-93%. For a Left-Skew (columns 6-7 in Table 2), again LM outperforms MAJ for every \(k\). The trend changes for the Right-Skew (columns 2-3). Here MAJ outperforms LM. All of these results were found significant with a confidence level of 95% using a one-tailed t-test. The results indicate that it is advised to consider the users assumed behavior. If the user preferences are assumed to be skewed towards some favorable item, as is the case with the Netflix dataset and with the simulated data with a right skew, it is best to use a heuristic that employs the Majority based strategy. If the results are uniformly skewed or it is assumed that some item or items are less preferred (left skew), it is best to choose a heuristics that employs the Least Misery strategy. Reducing the preference elicitation effort can also be achieved by changing the preference elicitation termination condition and requiring a winning item within top \(k\) instead of a definite item.

### 6. CONCLUSIONS

We propose an iterative preference elicitation method, where users are required to provide item ratings per request. We suggest a heuristic that attempts to minimize the preference elicitation effort under two aggregation strategies. The goal is to output top-\(k\) preferred items, where one of the items is a necessary winner, out of the top-\(N\) recommendations provided to the group by the recommender system (\(k < N\)). We show that the aggregation strategy affects the effort required in the preference elicitation and evaluate two state-of-the-art aggregation strategies. We demonstrate that the preference elicitation effort can be cut in up to 90% in some cases, while a necessary winner is still outputted. Since different aggregation strategies are used in different situations, in the future we plan to investigate other aggregation strategies and heuristics that combine a few strategies.

### 7. REFERENCES


