Reducing preference elicitation in group decision making

Lihi Naamani-Dery, Meir Kalech, Lior Rokach, Bracha Shapira

Abstract

Groups may need assistance in reaching a joint decision. Elections can reveal the winning item, but this means the group members need to vote on, or at least consider all available items. Our challenge is to minimize the amount of preferences that need to be elicited and thus reduce the effort required from the group members. We present a model that offers a few innovations. First, rather than offering a single winner, we propose to offer the group the best top-k alternatives. This can be beneficial if a certain item suddenly becomes unavailable, or if the group wishes to choose manually from a few selected items. Secondly, rather than offering a definite winning item, we suggest to approximate the item or the top-k items that best suit the group, according to a predefined confidence level. We study the tradeoff between the accuracy of the proposed winner item and the amount of preference elicitation required. Lastly, we offer to consider different preference aggregation strategies. These strategies differ in their emphasis: towards the individual users (Least Misery Strategy) or towards the majority of the group (Majority Based Strategy). We evaluate our findings on data collected in a user study as well as on real world and simulated datasets and show that selecting the suitable aggregation strategy and relaxing the termination condition can reduce communication cost up to 90%. Furthermore, the commonly used Majority strategy does not always outperform the Least Misery strategy. Addressing these three challenges contributes to the minimization of preference elicitation in expert systems.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

A group of people wishing to reach a joint decision faces the task of selecting the alternative that best suits the group out of all available candidate items. When all users' preferences are known, some voting aggregation strategy is used to compute and output the winning item to the group (Rossi, Venable, & Walsh, 2011). When the preferences are not available, a preference elicitation process is required.

Preference elicitation requires time and effort, so our goal is to stop the elicitation as soon as possible. In the worst case, for most voting protocols all the preferences are needed in order to determine a winning item, i.e., an item that most certainly suits the group's joint preferences (Conitzer & Sandholm, 2005). Nevertheless, in practice it has been shown that the required information can be cut in more than 50% (Kalech, Kraus, Kaminka, & Goldman, 2011; Lu & Boutilier, 2011). Given partial preferences, it is possible to define the set of the necessary winners, i.e., items which must necessarily win, as well as the set of possible winners, i.e., items which can still possibly win (Konczak & Lang, 2005). Using these definitions the elicitor can determine whether there is need for more information concerning the voters' preferences. Previous studies provide algorithms for preference elicitation of a single winner under the Range and the Borda protocols (Lu & Boutilier, 2011; Naamani-Dery, Golan, Kalech, & Rokach, 2015; Naamani-Dery, Kalech, Rokach, & Shapira, 2014). In this paper we define two tradeoffs that enable less elicitation: Selection and Approximation. Furthermore, we propose to examine different preference aggregation techniques.

Selection: a tradeoff exists between the amount of items outputted to the group and the cost of preferences elicitation required. Less elicitation effort is required for outputting k items where one of them is the winner with a high probability (top-k items) than for outputting one necessary winner (i.e., k = 1). Although outputting a definite winner is the most accurate result, there are advantages to outputting the top-k items. Not only is the communication cost reduced, it may actually be preferred to present a few alternatives to the user since if one of the alternatives is unavailable the group members can quickly switch to another already recommended alternative without requiring more elicitation (Baldiga & Green, 2013; Lu & Boutilier, 2010). Consider, for example, a setting of 30 optional dinner locations for a group. If a fish restaurant is the winning item, but one of the group members dislikes fish,
the group might prefer to switch to a different alternative rather than to perform another elicitation round.

**Approximation:** a different tradeoff is the one that exists between the accuracy of the proposed winner item and the amount of preference elicitation required. We suggest outputting an item that approximately suits the group with some confidence level rather than outputting an item that definitely suits the group. As we later show, the confidence level is based on the items’ winning probabilities. To reduce the elicitation even further, the two methods can be combined and top-k approximate items can be offered to the group. Consider, for example, a group that wishes to choose a movie to watch together out of movies available in the cinema. The members set the amount of options they wish to receive (k) and the level of confidence of the results. Thus, we define a new preference elicitation termination condition: approximate k-winner termination, namely where k items are found and one of them is the best item with a confidence level of $1 - \alpha$ ($0 \leq \alpha \leq 1$).

**Aggregation:** Ideally, the preference aggregation strategy (i.e., the voting protocol) should be a fair one. In his well-known work, Arrow shows that there is no perfect aggregation system (Arrow, 1951). One of the major differences between aggregation strategies is the social environment in which they are used; in particular, the perspective in which fairness is viewed. The emphasis can be either towards the individual user or towards the majority of the group (Jameson & Smyth, 2007). Two aggregation strategies that differ in their emphasis and are used in group recommender systems are the Majority Based Strategy and the Least Misery Strategy (Masthoff, 2011). Similar concepts can be found in the social choice literature, termed utilitarianism and egalitarianism (Myerson, 1981). In the Majority Based Strategy the users’ ratings of the different items are aggregated and the items with the highest total value are recommended. In the Least Misery Strategy the chosen items cannot be the least preferred by any of the users. The idea is that a group is as happy as its least happy member. One of the contributions of this paper is in proposing an efficient iterative preference elicitation algorithm which fits these strategies.

Overall, our goal is to reduce the communication cost in the preference elicitation process. We define the communication cost as the cost of querying one user for her preferences for one item. We allow users to submit the same rating for many items and do not request the users to hold a strict set of preferences over items. In this paper, we adopt the Range voting protocol which is adequate for this purpose; it requires users to submit a score within a certain range. Users are familiar with applications that ask for their score on an item, such as Amazon (www.amazon.com) or Netflix (www.netflix.com).

Preference elicitation becomes more challenging and interesting when a rating distribution of the voter-item preferences exists, i.e., a prior probability distribution of each voter’s preferences for each item. For example, in the case of a group of users wish to watch a movie together, the distribution can be inferred from rankings of these movies by similar users using collaborative filtering methods (Koren & Sill, 2011). After each user-item query, new information is revealed. The necessary and possible winner sets are updated to check whether or not the termination condition has been reached.

In this paper, we offer three main innovations contributing to the minimization of preference elicitation:

1. **Selection:** we suggest terminating preference elicitation sooner by returning k alternatives to the group members rather than returning just one item.

2. **Approximate winners:** we suggest computing approximate winner or winners with some confidence level. This as well reduces the communication cost.

3. **Aggregation:** we suggest considering the Least Misery aggregation Strategy beyond the known Majority based strategy.

We evaluated the approach on multiple datasets in different scenarios and application domains: (1) Two datasets that were collected using a group recommender system named “Lets Do It” which was built and operated in Ben-Gurion University. (2) Two real world datasets, the Netflix data (http://www.netflixprize.com) and Sushi data (Kamishima, Kazawa, & Akaho, 2005). (3) Simulated data which allow us to study the impact of the probability distribution. We show that selecting the suitable aggregation strategy and relaxing the termination condition can reduce communication up to 90%.

This paper is an extension of the authors’ previous short paper (Naamani-Dery, Kalech, Rokach, & Shapira, 2014). In the previous paper, we shortly presented one preference elicitation algorithm (DIG) without approximation. In this paper we added an overview of the state of the art in the field of voting techniques (Section 2). We extended the model and definitions and added a model for approximation of the necessary winner (Section 3). We added another algorithm, ES (Section 4). This allows us to compare the algorithms’ performance in different settings and show that each algorithm has an advantage in different scenarios. We have extended our evaluation to include a user-study, detailed experiments and a thorough analysis (Sections 5 and 6).

2. Related work

Group decision making consists of two phases: preference elicitation and preference aggregation. We start with describing the preference elicitation strategies considered in this paper, and move on to describe how preference elicitation.

2.1. Preference aggregation strategies

One of the contributions of this paper is to consider the Least Misery strategy, which, to our best knowledge, has not been studied in the context of preference elicitation. Throughout this paper we use “Majority” and “Least Misery” to refer to the Majority based strategy and the Least Misery based strategy (Masthoff, 2004).

Different studies have shown how different strategies affect group members (Masthoff, 2004; Senot, Kostadinov, Bouzid, Picault, & Aghasaryan, 2011). Masthoff studies how humans prefer to integrate personal recommendations. She concludes that users use the Majority Strategy, the Least Misery strategy and Majority without Misery strategy (Masthoff, 2004). Her findings motivate our research to focus on the Majority and the Least Misery strategies. These two strategies were also chosen by Baltrunas, Makcinskius, and Ricci (2010), in research focusing on the evaluation of the effectiveness of Group Recommender Systems obtained by aggregating user preferences.

In the Majority Strategy the users’ ratings of the different items are aggregated and the item with the highest total value is the winner. Note that the result is similar to selecting the item with the highest average, thus this strategy is sometimes referred to as the Average Strategy or the Additive Strategy (Masthoff, 2011). The Majority strategy is used in numerous applications. For example: in the MusicFX system the square of the individual preferences are summed (McCarthy & Anagnost, 1998) and the Travel Decision Forum assists in planning a holiday (Jameson, 2004). Yet another example is of TV programs recommendation for a group (Masthoff, 2004; Yu, Zhou, Hao, & Gu, 2006), where the chosen program fits the wishes of the majority of the group. A disadvantage of this strategy is that it can be unfair towards users with the minority view. In fact, Yu et al. (2006) state that their system works well for a homogenous group but when the group is heterogeneous, dissatisfaction of the minority group occurs.
The Least Misery Strategy defines that the chosen items cannot be the least preferred by any of the users. In the Polylens system the Least Misery strategy is used to recommend movies to small groups (O’connor, Cosley, Konstan, & Riedl, 2002). Survey results show that 77% of the users found the group recommendation more helpful than the personal one. The disadvantage is that the minority opinion can dictate the decision for the entire group – if all the users of the group except one prefer some item, it is still not chosen (Masthoff, 2011).

2.2. Preference elicitation

Voting with partial information, i.e., when voters do not reveal their preferences for all candidates, has a theoretical basis (Conitzer & Sandholm, 2005; Konczak & Lang, 2005). Conitzer and Sandholm (2005) analyze the communication complexity of various voting protocols and determine upper and lower bounds for communication costs (communication with the users). In general, they show that for most voting protocols, in the worst possible case voters should send their entire set of preferences. Konczak and Lang (2005) demonstrate how to compute the set of possible winners and a set of necessary winners. These sets determine which candidates no longer have a chance of winning and which will certainly win. We adopt their approach and propose a system where the agents do not need to send their entire set of preferences.

A theoretical bound for the computation of necessary winners has been addressed (Betzler, Hemmann, & Niedermeier, 2009; Pini, Rossi, Venable, & Walsh, 2009, 2007) and so has the theoretical complexity of approximating a winner (Service & Adams, 2012). The complexity of finding possible winners has also been investigated (Xia & Conitzer, 2008). Others considered settings where preferences may be unspecified, focusing on soft constraint problems (Gelain, Pini, Rossi, & Venable, 2007) or on sequential majority voting (Lang, Pini, Rossi, Venable, & Walsh, 2007). They do not provide empirical evaluation nor do they focus on reducing the communication load. Ding and Lin (2013) define a candidate winning set as the set of queries needed in order to determine whether the candidate is a necessary winner. The authors show that for rules other than Plurality voting, computing this set is NP-hard. Following this theorem, we propose heuristics for preference elicitation.

Prior probability distribution of the votes is assumed by Hazon, Aumann, Kraus, and Wooldridge (2008). The winning probability of each candidate is evaluated using Plurality, Borda and Copeland protocols (Brams & Fishburn, 2002). Hazon et al. also show theoretical bounds for the ability to calculate the probability of an outcome. Bachrach, Betzler, and Faliszewski (2010) provide an algorithm for computing the probability of a candidate to win, assuming a voting rule that is computable in polynomial time (such as range voting) and assuming a uniform random distribution of voters’ choice of candidates. However, while both Hazon et al. and Bachrach et al. focus on calculating the winning probability for each candidate, we focus on practical vote elicitation and specifically on approximating the winner within top-k items using a minimal amount of queries.

Practical vote elicitation has been recently addressed using various approaches. Pfeiffer, Gao, Mao, Chen, and Rand (2012) predict the ranking of n items; they elicit preferences by querying voters using pairwise comparison of items. However, they do not explicitly aim to reduce the number of queries. Furthermore, they assume that each voter can be approached only once and that there is no prior knowledge about the voters. As a result, voter-item distributions cannot be computed. Their method is therefore suitable when a large amount of voters is available and the task is to determine some hidden truth (also known as the wisdom of the crowd).

We, on the other hand, wish to reach a joint decision for a specific group of voters. Chen and Cheng (2010) allow users to provide partial preferences at multiple times. The authors developed an algorithm for finding a maximum consensus and resolving conflicts. However, they do not focus on reducing the preference elicitation required. Another approach is to require the voters to submit only their top-k ranked preferences and these are used to predict the winning candidate with some probability (Filimus & Oren, 2014; Oren, Filmus, & Boutilier, 2013). This usage of top-k is different than ours; we ask users to submit their rating for certain chosen items, and output top-k alternatives to the group. All of these methods address vote elicitation, however they do not try to minimize the amount of queries the users receive.

An attempt to reduce the number of queries is presented by Kalech et al. (2011). They assume that each user holds a predefined decreasing order of the preferences. In an iterative process, the voters are requested to submit their highest preferences; the request is for the rating of a single item from all the users. One major disadvantage of this approach is that requiring the users to predefine their preferences can be inconvenient to the users. Another practical elicitation process is proposed for the Borda voting protocol using the minmax regret concept. The output is a definite winner or an approximate winner, but the approximation confidence level is not stated (Lu & Boutilier, 2011). The method was later extended to return multiple winners, again using the Borda protocol and minmax regret (Lu & Boutilier, 2013). Two practical elicitation algorithms that aim to minimize preference communication have been presented for the Range voting protocol (Naaman-Dery et al., 2014). In this paper, we use these algorithms as the basis to address the presented challenges. We iteratively query one voter for her preferences regarding one item.

The advantage of our approach is that users are not required to predefine their preference as in Kalech et al. (2011) and are not required to hold a strict set of ordered untied preferences as in Lu and Boutilier (2013). Also, in previous research the Majority strategy is used for preference aggregation. To the best of our knowledge, the issue of preference elicitation and returning one or more items under the Least Misery strategy has not yet been investigated, although Least Misery is favored by users (Masthoff, 2004). Furthermore, we have not encountered any study that approximates a winner or k alternative winner with a confidence level. We present algorithms which can use the Majority or the Least Misery strategy in order to output one or top-k definite winner items or approximate winner items within some confidence level.

3. Model description

We introduce a general approach for reaching a joint decision with minimal elicitation of voter preferences. We assume that the users’ preferences are unknown in advance, but can be acquired during the process, i.e., a user who is queried for her preference for an item will answer the query and rate the item in question. We also assume that users submit their true preferences. Therefore, in this paper we do not handle manipulations or strategic voting.

3.1. Problem definition

Let us define a set of users (voters) as $V = \{v_1, v_2, \ldots, v_m\}$ and a set of candidate items as $C = \{c_1, c_2, \ldots, c_n\}$. We define a request for specific information from the voter as a query $q$. A query has a cost, e.g., the cost of communication with the voter, or the cost of interfering with the voter’s regular activities.

Definition 1. (Cost): Given a query $q \in Q$, the function cost : $Q \rightarrow \mathbb{R}$ returns the communication cost of the query.
Throughout this paper, we assume that the cost is equal for all queries. It is possible to determine the winner from partial voters’ ratings (Konczak & Lang, 2005; Walsh, 2007). We adopt an iterative method (Kalech et al., 2011), which proceeds in rounds. In each round one voter is queried for her rating of one item. Consequently, we determine the next query, such that the total expected cost is minimized.

Let $O^i$ represent the set of voter $v_i$’s responses to queries. Note that this set does not necessarily contain all the items, since the voter might have not been queried regarding some of the items, and therefore has not rated them. All the preferences of all the voters are held in $O^j$, $O^j = \{O^1, \ldots, O^m\}$ is a set of $O^i$ sets. At the end of each round, one query response is added to $O^j$. When the set of voters’ preferences $O^j$ contains enough (or all) preferences, the winning item can be computed using some preference aggregation strategy. Our goal is to guarantee or approximate the winner with minimal cost.

**Definition 2.** (Winner with minimal cost problem): Let $O^j_1, \ldots, O^j_s$ represent possible sets of queries and let $R^j_1, \ldots, R^j_s$ represent the aggregated cost of the queries in each set respectively. A winner with minimal cost is the winner determined using the set with the lowest cost: $\text{argmin}_j (R^j_1)$.

3.2. The framework

We assume the existence of a voting center whose purpose is to output items given the termination conditions: top-$k$ items where one of the items is the winner with $1 - \alpha$ confidence level. These items are presented to the group as their best options.

The complete process is illustrated in Fig. 1 and proceeds as follows: A distribution center holds a database of historical user and item ratings, and the users’ social networks graph if it is available. These ratings (with the possible addition of the graph) are used to compute voter-item probability distributions. A voting center receives the voter-item probability distributions from the distribution center and is responsible to execute one of the heuristics described in Section 4. The heuristics outputs a voter-item query, i.e., a request for one specific voter to rate one specific item. This query is sent to the appropriate voter. Once the voter responds, the voting center checks whether there are enough preferences to stop the process. This is done by examining the value of the threshold for termination. If reached, the process terminates. If not, the voting center resumes charge. The user’s response is sent to the distribution center so that the probability distributions can be updated. The voting center outputs a new query and the process repeats until the termination condition is reached. The termination condition depends on the preset confidence level $1 - \alpha$ ($0 \leq \alpha \leq 1$) and on $k$, the number of outputted items. For example, when $\alpha = 0$ and $k = 1$, the process terminates once one definite item is found.

When queried, the voters assign ratings to the items from a discrete domain of values $D$ where $d_{\text{min}}$ and $d_{\text{max}}$ are the lowest and highest values, respectively. User $v_i$’s preferences are represented by the rating function value: $V \times C \rightarrow D$.

We assume that there exists a prior rating distribution of the voter-item preferences, i.e., a prior distribution of each voter’s preferences for each item. The distribution can be inferred from rankings of similar users using collaborative filtering methods (Koren & Sill, 2011) or using the user’s social networks graph, if such exists (Ben-Shimon et al., 2007). In order to decide on the next query, the elictor considers the rating distribution of the voter-item preferences. We denote $q^j_c$ as the single preference of $v_i$ for a single item $c$. The rating distribution is defined as follows:

**Definition 3.** (Rating Distribution): the voting center considers $q^j_c$ as a discrete random variable distributed according to some rating distribution $\nu d^j_c$, such that $\nu d^j_c [d_k] = P_3 (q^j_c = d_k)$.

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

We assume independence between the probability distributions of each voter. 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 to realistically hold, since each distribution may depend on each of the other distributions. When facing the tradeoff between the model’s 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 machine learning field, a similar naive assumption is known to provide accurate classification, even though the independence assumption is not always true (Domingos & Pazzani, 1997). We therefore argue that the system’s loss of accuracy, if at all exists, is insignificant.

The initial estimated rating distribution is calculated in advance (Koren & Sill, 2011). The estimated distribution is then updated every time a new rating is added. The accuracy is expected to grow with the number of ratings acquired.

In the next sections we present the aggregation strategies (Section 3.3), the termination conditions (Section 3.4) and the elicitors (Section 4).

### 3.3. Aggregation strategies

The item's score depends on the strategy used. Throughout this paper, we denote the employed aggregation strategy \(str\). We now define the aggregation strategies: Majority and Least Misery. As mentioned, in the Majority strategy the emphasis is towards the majority of the group:

**Definition 4.** (Majority Strategy): given the users’ preferences, the Majority Strategy computes the score of item \(c_j\) as follows:

\[
s_{\text{majority}}(c_j) = \sum_{i \in \{1 \ldots m\}} q^i_j
\]

In the Least Misery strategy, the chosen item cannot be the least preferred by any of the users.

**Definition 5.** (Least Misery Strategy): given the users’ preferences, the Least Misery Strategy computes the score of item \(c_j\) as follows:

\[
s_{\text{least}}(c_j) = \min_{i \in \{1 \ldots m\}} q^i_j
\]

Each of the two strategies has its pros and cons. The choice of the strategy might impact the result. Consider the example in Table 2, showing the preferences of three users for 3 items. According to the Majority strategy, the winning item is item \(c_1\), with a total score of 11, followed by items \(c_3\) and \(c_2\). According to the Least Misery strategy, the winning item is \(c_2\), with a score of 3.

### 3.4. Termination conditions

During the preference elicitation process, the preferences are submitted to the voting center and the voting center aggregates the preferences. This process continues until a termination condition is reached. The termination condition is pre-set by the system administrator according to the group’s request. The termination condition is one of the following: a definite winning item, an approximate winning item with some confidence level, top-\(k\) items where one of them is the winner, or approximate top-\(k\) items where one of the items is the winner with some confidence level.

Given a set of responses to queries and a termination condition, the goal is to determine whether the iterative process can be terminated. Let \(O^j = \{q^1_j, \ldots, q^m_j\}\) represents the set of voter \(v_j\)’s responses to queries. Note that this set does not necessarily contain all the items. \(O^A = \{O^1, \ldots, O^m\}\) is a set of \(O^j\) sets. The function \(pmx^A(c_j, O^A)\) computes the possible maximum rating for item \(c_j\), given the known preference values of the voters.

**Definition 6.** (Possible Maximum): given the set of responses \(O^A\) and an aggregation strategy \(str\), the possible maximum score of candidate \(c_j\), denoted \(pmx^A(c_j, O^A)\), is computed as follows:

\[
pxm^A(c_j, O^A, str) = \begin{cases} \sum_{i \in \{1 \ldots m\}} pmx^i(c_j, O^i) & \text{str} = \text{majority} \\ \min_{i \in \{1 \ldots m\}}(pmx^i(c_j, O^i)) & \text{str} = \text{least misery} \end{cases}
\]

where \(pmx^i(c_j, O^i) = \begin{cases} d_g & \text{if } q^i_j = d_g \\ d_{\max} & \text{otherwise} \end{cases}\)

Similarly, the function of the possible minimum rating of item \(c_j\) \(min^A(c_j, O^A)\) is:

**Definition 7.** (Possible Minimum): given the set of responses \(O^A\) and an aggregation strategy \(str\), the possible minimum score of candidate \(c_j\), denoted \(min^A(c_j, O^A)\), is computed as follows:

\[
pxm^A(c_j, O^A, str) = \begin{cases} \sum_{i \in \{1 \ldots m\}} pmn^i(c_j, O^i) & \text{str} = \text{majority} \\ \min_{i \in \{1 \ldots m\}}(pmn^i(c_j, O^i)) & \text{str} = \text{least misery} \end{cases}
\]

where \(pmn^i(c_j, O^i) = \begin{cases} d_u & \text{if } q^i_j = d_u \\ d_{\min} & \text{otherwise} \end{cases}\)

Consider the example given in Table 1, but assume that the rating of \(c_1\) for \(v_3\) is unknown: thus, \(q^3_1 = 5\), \(q^3_2 = 4\), \(q^3_3 = ?\). Using Definitions 6 and 7 we can compute the possible maximum and minimum under each aggregation strategy.

### 3.4.1. Selection among top-\(k\) alternatives

One possible termination condition is to stop the preference elicitation process once at least one winner is found (Kalech et al., 2011; Lu & Boutiller, 2013; Naamani-Dery et al., 2014). We follow Kalech et al. (2011); Lu and Boutiller (2011) and define a necessary winner set (NW) as a set of items whose possible minimum aggregated rating is equal or greater than the possible maximum aggregated rating of all the others\(^1\):

\(^1\) Konczak and Lang (Konczak & Lang, 2005) affirm this definition of necessary and possible winners in proposition 2 of their paper.
Definition 8. (Necessary Winners Items Set): 
\[ \text{NW} = \{ c_j | p_{max}^A(c_j, O^A) \geq p_{max}^A(c_i, O^A) \, \forall c_i \in \mathcal{C} \setminus \{ c \} \} \]

We assume that there is only one necessary item, although the necessary winner set may theoretically contain more than one item. Note that for a large number of voters, in most cases there is just one winner item. In the case of more than one winning item, the first item is selected lexicographically.

In some cases, the group members can be satisfied with a shorter preference elicitation process. They may agree to trade the result accuracy with less elicitation cycles. In other words, instead of terminating the preference elicitation once a necessary winner is found, the group may agree to terminate the preference elicitation once a set of top-k items is presented to them. One of these items is the necessary winner, but without further elicitation it is not possible to determine which of the items it is. To accurately define the top-k items, let us define first the possible winner group. The possible winners are all the items whose possible maximum aggregated rating is greater than or equal to the possible minimum rating of all the other items.

Definition 9. (Possible Winners Set): 
\[ \text{PW} = \{ c_j | p_{max}^A(c_j, O^A) \geq p_{min}^A(c_i, O^A) \, \forall c_i \in \mathcal{C} \setminus \{ c \} \} \]

Note that the possible winning group subsumes the necessary winners: \( \text{NW} \subset \text{PW} \). After each query, the necessary winner set and the possible winner set need to be recalculated. To begin with, when none of the preferences are known, the possible winner set contains all items: \( |\text{PW}| = |\mathcal{C}| \) and the necessary winner’s set is empty: \( |\text{NW}| = 0 \). The process is terminated once the size of the set of possible winners is reduced to \( k \). We denote the necessary winners set of size \( k \) and the possible winners set of size \( k \): \( \text{NW}^k \) and \( \text{PW}^k \) respectively. Thus, the set contains the top-k possible winners, where, by definition, these top-k are guaranteed to include the necessary winners.

3.4.2. Winner approximation

We examine the accuracy-elicitation tradeoff. The preference elicitation process can be reduced, but the accuracy of the output is affected: the returned items are estimated to contain the winning item at some confidence level, with an error rate \( \alpha \). To compute a winner with some confidence level we should first define the score function of the aggregation. The score \( s \) the candidate can achieve after aggregating the preferences of the voters depends on the strategy:

\[ S = \begin{cases} 
\left( n \cdot d_{\min} - n \cdot d_{\min} + 1 \ldots n \cdot d_{\max} \right) & \text{if } str = \text{majority} \\
\left( d_{\min} \cdot d_{\min} + 1 \ldots d_{\max} \right) & \text{if } str = \text{least} 
\end{cases} 
\]

Let us begin by examining the probability that one item has a certain score: \( Pr(c_j = s) \). The probability of any item to be the winner is:

Definition 10. (Item Winning Probability): Under the independence of probabilities assumption, the probability that item \( c_j \) is the winner is the aggregation of \( c_j \)'s probabilities to win over the possible ratings \( s \):

\[
Pr(c_j \in \text{NW}) = \sum_{s \in \mathcal{S}, \forall j \neq j} Pr(c_j = s | v_1, \ldots, v_m) \cdot Pr(c_j < s)
\]

The probability that given \( m \) voters an item will receive the score \( s \) \( Pr(c_j = s | v_1, \ldots, v_m) \) can be computed recursively. This probability depends on the aggregation strategy. For the Majority strategy we use:

\[
Pr(c_j = s | v_1, \ldots, v_m) = \sum_{x=d_{\max}}^{d_{\min}} \left( Pr(c_j = s - x | v_1, \ldots, v_{m-1}) \cdot Pr(q_m = x) \right)
\]

where \( Pr(c_j = s | v_1) = Pr(q_1 = s) \)

For the Least Misery strategy we use:

\[
Pr(c_j = s | v_1, \ldots, v_m) = \sum_{x=d_{\max}}^{d_{\min}} \left( Pr(c_j = s | v_1, \ldots, v_m) \cdot Pr(q_m = x) \right)
\]

\[
+ \sum_{x=d_{\max}+1}^{d_{\max}} \left( Pr(c_j = s | v_1, \ldots, v_m) \cdot Pr(q_m = s) \right)
\]

In both strategies we compute the probability that an item will receive a score of at most \( s \) as follows (\( s \) is defined in Eq. (1)):

\[
Pr(c_j < s) = \sum_{x=d_{\min}(s)}^{s-1} Pr(c_j = x | v_1 \ldots v_m)
\]

The following is a step by step running example, for the Majority strategy for \( d = \{1, 2, 3\} \). The example is based on the voting distributions (VD’s) presented in Table 1: note that \( Pr(q_1^3 = 3) = 0.4 \), \( Pr(q_2^3 = 2) = 0.3 \), \( Pr(q_3^3 = 1) = 0.3 \). We start by calculating \( Pr(c_j = s) \). The calculation is done using a dynamic programming algorithm where each result is calculated using the previously calculated results. For instance, using Eq. (2), \( Pr(c_1 = 6) \) based on the ratings of voters \( v_1, v_2, v_3 \):

\[
Pr(c_1 = 6 | v_1, v_2) = Pr(c_1 = 5 | v_1, v_2) \times Pr(q_1^3 = 1) + Pr(c_1 = 4 | v_1, v_2) \times Pr(q_1^3 = 2) + Pr(c_1 = 3 | v_1, v_2) \times Pr(q_1^3 = 3).
\]

In the same manner: \( Pr(c_1 = 5 | v_1, v_2) = 0.14 \), \( Pr(c_1 = 4 | v_1, v_2) = 0.36 \), \( Pr(c_1 = 3 | v_1, v_2) = 0.24 \) so that finally \( Pr(c_1 = 6 | v_1, v_2, v_3) = 0.236 \). Next, we calculate \( Pr(c_1 \leq s) \) using Eq. (4): \( Pr(c_1 < 6) = Pr(c_1 = 3) + Pr(c_1 = 4) + Pr(c_1 = 5) \).

To define top-k with a confidence level we define first PV as a vector of items, ordered according to their winning probabilities:

Definition 11. (Ordered Vector of winning probabilities): PV is an array of decreasingly ordered items according to their winning probabilities.

The probability that the winner is within the top-k is actually the aggregated winning probabilities of the first \( k \) items in PV. The more preferences elicited from the users the higher the probability the winner is within the top-k. The confidence level is a value which determines an upper bound for the probability of the winner to be among the top-k. The preference elicitation process is terminated once the confidence level equals \( 1 - \alpha \). Formally, the termination condition is:

Definition 12. (Termination with top-k approximate items): the preference elicitation process terminates for a given \( k \) and \( \alpha \), when \( \sum_{i=1}^{k} PV[i] \geq 1 - \alpha \) where \( 0 \leq \alpha \leq 1 \).

4. Elicitors

The elicitor selects the next query according to one of two heuristics, DIG or ES.

4.1. Entropy based heuristic

The Dynamic Information Gain (DIG) Heuristic is an iterative algorithm (Naamani-Dery, Kalech, Rokach, & Shapira, 2010). 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 expected information gain. The expected information gain of a specific query is influenced by 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 with a confidence level of $1 - \alpha$. In order to select a query, the information gained from each one of the optional queries is calculated and then the query one that maximizes the information gain is selected. To compute the information gain, the winning probability of each item is calculated. Next, the information gain of the $m \times n$ possible queries is computed. The expected information gain of a query is the difference between the prior expected entropy and the posterior expected entropy, given the possible responses to the query. The entropy of the necessary winner within top-$k$ ($NW^k$) is computed as follows:

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

**Definition 13.** (Information Gain): The Expected Information Gain ($IG$) of a query $q^j$ is:

$$IG(NW^k|q^j) = H(NW^k) - \sum_{g=\min}^{\max} H(NW^k|q^j = d_g) \cdot Pr(q^j = d_g)$$

where $H(NW^k|q^j = d_g)$ represents the expected entropy of $NW^k$ given the possible values by querying voter $v_i$ on item $c_j$.

The query that maximizes the information gain is selected: $\arg\max IG_{ij}(NW^k|q^j)$. The query selection process continues until the termination condition is reached, i.e., once a winner within top-$k$ items is found with $1 - \alpha$ confidence. Note that the termination condition is determined by $\alpha$ and $k$. However, the termination condition does not affect the systems information gain.

4.2. Expected maximum based heuristic

The highest expected heuristic (ES) score is based on the exploration vs. exploitation tradeoff (Naamani-Dery et al., 2010). As mentioned earlier, a necessary winner is an item whose possible minimum is greater than the possible maximum of the other items. The possible maximum of an item decreases while its possible minimum increases as more information about voter preferences is revealed. Thus, an item for which no voter has yet submitted a rating has the highest possible maximum and must be considered as a possible winner. On the other hand, such an item also has the lowest possible minimum and cannot yet be a winner. Therefore, for more information, we may want to explore the voters’ preferences for the items in order to determine their potential of being a winner within top-$k$. Once we have enough information about the items’ rating, we can exploit this information to further inquire about the items that are more likely to win.

We propose a heuristic which chooses its next query by considering the item that has the possible maximum and the voter that is expected to maximize the rating of that item. The expected rating of $q^j$ based on the rating distribution $vd^j$ is:

$$ES(vd^j) = \sum_{g=\min}^{\max} Pr(q^j = d_g) \cdot d_g$$

(5)

For item $c_j$, we choose the voter that maximizes the expected rating: $\arg\max ES(vd^j)$. Using this approach, we encourage a broad exploration of the items since the less information we have about an item’s rating, the higher possible maximum it has. In addition, we exploit the preferences revealed in order to: (1) refrain from querying about items that have been proven as impossible winners (since their possible maximum is less than a minimum of another item); (2) further examine an item that has the highest possible maximum and might be a necessary winner.

The following is an illustration of the algorithm using the example employed in the previous section. To begin with, we have only probabilistic knowledge of voter preferences. Since no voter has submitted any preference yet, in the first round the possible maximum of each item is nine (since there are three voters and the maximum rating that can be assigned is three). The first item $c_1$ is selected for a query according to the tie breaking policy. According to the distribution in Table 1, the expected ratings of the voters over $c_1$ are:

$$ES(vd^1) = 0.2 \cdot 1 + 0.2 \cdot 2 + 0.6 \cdot 3 = 2.4$$

$$ES(vd^1) = 0.4 \cdot 1 + 0.3 \cdot 2 + 0.3 \cdot 3 = 1.9$$

$$ES(vd^1) = 0.3 \cdot 1 + 0.3 \cdot 2 + 0.4 \cdot 3 = 2.3$$

Thus, the voter-item query pair is $q^1$. Assuming the voter’s response is $q^1 = 2$, in the next iteration the possible maximum of $c_1$ is 8 and of $c_2$ is 9. Therefore, in the next round, $c_2$ is selected as the item in the voter-item query pair. The algorithm iterates until the termination condition is reached. For instance, if the termination condition is an approximate item with $\alpha = 0.05$ and $k = 3$, the algorithm will repeat until a necessary winner is one of the top-3 items, with a probability of 95%.

5. Evaluation

We first present the research questions and research procedure (Section 5.1), the datasets evaluated (Section 5.2) and then present the evaluation on: different top-$k$ termination conditions (Section 5.3), different confidence levels for approximation (Section 5.4), and a comparison of the two aggregation strategies (Section 5.5).

5.1. Research questions and research procedure

We present an empirical evaluation of the following research questions:

(a) **Selection** – To what extent does outputting top-$k$ items reduce the required number of queries (Section 5.3)?

(b) **Approximation** – there is a tradeoff between outputting an approximate winner, or approximate top-$k$ items and outputting a definite winner or definite top-$k$ items. To what extent does the approximation accuracy improve as more data is collected (Section 5.4)?

(c) **Aggregation** – How does the aggregation strategy affects the preference elicitation process? We examine two aggregation strategies: with emphasis towards the group and with emphasis towards the user; i.e., the Majority and Least Misery strategies (Section 5.5).

We examine the performance of the algorithms presented in Section 4: DIG and ES. As mentioned in the related works section, 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 of the RANDOM algorithm each experiment is repeated 20 times. We evaluate the methods in terms of:

(1) **Number of queries** - we measure the reduction in communication cost by measuring the number of queries required for finding a necessary or approximate winner.

(2) **Approximation accuracy** – the approximation accuracy is measured in two ways:

(a) Probability that the winner is within the top-$k$
(b) Confidence level accuracy - a confidence level refers to the percentage of all possible samples that can be expected to include the true population parameter. The confidence level \((1 - \alpha)\) is accurate, if the winner is indeed within the top-\(k\) items in \((1 - \alpha)\)% of the experiments.

In order to conclude which algorithm performs best over multiple datasets, we follow a robust non-parametric procedure proposed by (García, Fernández, Luengo, & Herrera, 2010). We first used the Friedman Aligned Ranks test in order to reject the null hypothesis that all heuristics perform the same. This test was followed by the Bonferroni-Dunn test to find whether one of the heuristics performs significantly better than other heuristics.

5.2. Datasets

We evaluated our methods on different domains:

- **Simulated datasets**: allowed us to manipulate the data and thus further study the different parameters.
- **Real-world datasets**: the Netflix data (http://www.netflixprize.com), Sushi dataset (Kamishima et al., 2005).
- **User Study datasets**: Pubs dataset and Restaurants dataset datasets from the “Let’s do it” 2 recommender system, a user study performed in Ben Gurion University during Spring 2014.

In each domain we considered a setting of a group of 10 members and 10 items. We chose to restrict the users-items matrix to 10 x 10 since we tried to model a common scenario of a small group of people that wish to reach a joint decision. It would be impractical to suggest an elicitation process on large numbers of items (e.g., 16,000 movies). Even for less extreme cases, studies have shown that too much choice can be demotivating. Users are more satisfied when presented with a single digit number of options to choose from Lyengar and Lepper (2000). Therefore in cases where a large number of items exist, we suggest to apply a preference elicitation voting procedure on the top \(N\) ranked items only. We assume that when more than \(N\) items are available, some recommender system can be used to provide a ranked list of all items.

Our focus is the analysis of the contribution of returning a winner within top-\(k\) items, thus narrowing down the top-\(N\) suggestions received by a recommender system \((k \leq N)\). An additional focus is on approximating a winner and on the aggregation strategies. The analysis of the scaling of the matrix sizes and the runtime has been evaluated in Naamani-Dery et al. (2014).

5.2.1. Simulated datasets

The first domain is a simulated meeting scenario where voters are required to vote for their preferred time slot for a meeting. Simulating the data allows us to investigate different distribution settings and a wider variety of scenarios than those given in one dataset. The users rate their preferences on a scale of 1–4. We manipulated the user-item distribution skewness, i.e., the measure of the asymmetry of a distribution. A higher absolute skewness level indicates a higher asymmetry. A negative skew indicates that the distribution is concentrated on high vote values while a positive skew indicates the distribution is concentrated on low vote values. Similarly to Naamani-Dery et al. (2014), we created user-item rating distributions with different skewness levels. We chose 3 extreme cases, as presented in Table 3: a user is in favor of the item (skewness level \((-6)\)), a user dislikes the item (skewness level 6), a user is indifferent (skewness level 0). In the experiments, the skewness level of one of the items is set in advance and all other items receive a uniform skew (skew “0” in Table 3). Having set a rating distribution for every user-item pair, we randomly sample from the distribution to set the voter-item rating. To account for randomness, each experiment was repeated 10 times.

5.2.2. Real-world datasets

The Netflix prize dataset (http://www.netflixprize.com) is a real world dataset containing the ratings voters assigned to movies. The dataset consists of \(~100,000\) users and \(~16,000\) items. We consider a setting of a group of 10 users and 10 items. The users in the group were drawn randomly from a subset of Netflix where all of the users rated 100 items and there were no missing values. 90 items were used to create the initial rating distribution as described in Naamani-Dery et al. (2014). The 10 remaining items were set as the items in question. To account for Randomness, 10 different groups were extracted in this manner.

The Sushi dataset (Kamishima et al., 2005) is a real world dataset that contains 5000 preference rankings over 10 kinds of sushi. 10 different groups of 10 different users each were drawn at random from the dataset. Since only 10 items exist in the dataset, the initial user-item probability distribution was set to uniform. The distribution was updated after each query. We examined a scenario of 10 users who have to decide between 10 sushi types, using subsets of the dataset. We derived 10 different random matrices from each scenario size.

5.2.3. User study

We created our own set of real data and examined two scenarios of a group that wishes to: (a) select a restaurant or (b) select a pub or club. The data was collected using a group recommender system, named “Let’s Do It”.

The system obtained a full set of ratings from 90 students in Ben Gurion University, for two different domains: (a) restaurants (16 items) and (b) pubs and clubs (23 items). Fig. 2 presents the opening screen. The students were instructed to rate each item on a 1 to 5 scale, according to their satisfaction from past visits, or in case they were unfamiliar with a place, according to how appealing it was for them to visit it. Each item had a picture and a short description, as shown in Fig. 3. The students could view the items they rated, the items left for them to rate. They could also change the ratings. This is demonstrated in Fig. 4. Rating distributions were derived in the same manner as for the Netflix dataset (Section 5.2.1).

5.3. Selection of top-\(k\) items

We examined different top-\(k\) 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 are for the Majority aggregation strategy with a 100% confidence level (\(\alpha = 0\)). Different confidence levels and a comparison between the performance of the Majority strategy and the Least Misery strategy are presented in the next sections. We first report the results of three levels of skewness of simulated data, followed by the real world datasets: Netflix, Sushi, Pubs, and Restaurants.

We examine three different skewness levels of simulated data. Figs. 5–7 present results for a skewness level of (6), (0) and (-6)
Fig. 2. The student rate pubs&clubs and restaurants.

Fig. 3. Rating for two clubs.

Fig. 4. The student can see what places need to be rate.
respectively. Axis x presents the termination conditions $k = 1, \ldots, 10$. Axis y presents the amount of queries required in order to terminate and find a winner within the top-k. A larger k means that the termination condition is relaxed and less queries are needed. Indeed, in all cases, as $k$ increases, the amount of queries decreases. The performance of RANDOM is not significantly affected by skewness levels. For a skewness level of $-6$ (Fig. 5), DIG outperforms ES and RANDOM and requires the least amount of queries. For a skewness level of $0$ and of $(6)$, ES outperforms DIG and RANDOM for the top-1 to top-3 items. Then, DIG resumes charge and provides better results (Figs. 6 and 7).

We now turn to examine the real world datasets. On the Netflix dataset (Fig. 8), the trend is similar to that obtained on the skewness level of 0 and 6. That is, for top-1 to top-3 ES is superior, and then DIG maintains the lead. Again, DIG displays a sharp curve while ES requires almost the same number of queries regardless of the termination point (the top-k). The same phenomenon is radicalized on the Pubs dataset (Fig. 10) and on the Restaurants dataset (Fig. 11); not only does DIG take the lead after the top-3 items, but also, Random exhibits better performance than ES (but not than DIG) when for more than top-6 items. On the Sushi dataset (Fig. 9) DIG outperforms ES and RANDOM for all k.

The results can be explained by considering the properties of the heuristics and of the datasets. In a setting of a simulated skewness of $-6$ the votes are skewed towards the winner and it is more obvious who the winner is. It is less obvious who the winner is when the skewness level is 0 or 6 in simulated data. Also, when $k$ is smaller, ES performs better, since ES is designed to seek for potential winning items. Therefore, the amount of queries ES requires is more or less constant regardless of the k items required for output. DIG is designed to focus on reducing entropy. When $k$ is larger the entropy reduces faster. In the Sushi dataset the initial user-item distribution is uniform so all items have the same chance to be the winning item. Thus, the initial state in the Sushi
dataset is similar to a simulated skewness data with (0). However, in the Netflix, Pubs, and Restaurants datasets the distributions are estimated and there is a skewness pattern which enables DIG to outperform. Furthermore, when it is less obvious who the winner is (as in Netflix), the differences in the heuristics performance are smaller.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 95% rejects the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG and ES significantly outperform RANDOM at a 95% confidence level.

5.4. Approximation

We examined the amount of queries required under different confidence levels (Figs. 12 and 13), when a definite winner (k = 1) is required. For the simulated data, we set the skewness level to neutral (0). The results presented here are for the Majority strategy, while a comparison between the two aggregation strategies is presented in the next section. We also examine the accuracy of the approximations.

Axis x presents the required confidence level: from 50% to 100% (100% is a definite winner). Axis y presents the amount of queries required in order to terminate and find the top-k items. For the simulated data, there is a steady increase in the required amount of queries (Fig. 12) for all heuristics. DIG outperforms ES and RANDOM, while RANDOM is the least performer (Fig. 14). The steady increase in the amount of queries for the simulated dataset and for the Sushi dataset (Kamishima et al., 2005), Pubs dataset (Fig. 15) and Restaurants dataset (Fig. 16) can be easily explained since more queries are needed in order to gain more information for a higher accuracy level. However, the results for the Netflix dataset behave differently and require a deeper explanation.

For the Netflix data (Fig. 13), the increase in the required amount of queries is small for confidence levels 50%-95%. However, there is a big jump in the required number of queries when the desired confidence is 100% (a definite winner is required): from ~10 required queries to achieve a confidence level of 95%, to ~90 queries for a 100% confidence. The probability distributions for the Netflix dataset are estimated, whereas for the simulated data we have accurate (simulated) distributions. We show the probabilities accuracy for the datasets: simulated data with skewness level (0), Netflix and Sushi in Figs. 17, 18 and 19 respectively. Axis x is the iteration number and axis y is the probability that the winner is indeed within the top-k items. In this case, k = 1. For the simulated data (Fig. 17) the probability accuracy increases steadily as more information, acquired in the iterations, becomes available. On the other hand, since the Netflix, Pubs and Restaurants

---

**Fig. 12.** Approximations with simulated data with skewness (0).

**Fig. 13.** Approximations on the Netflix dataset.

**Fig. 14.** Approximations on the Sushi dataset.

**Fig. 15.** Approximations on the Pubs dataset.

**Fig. 16.** Approximations on the Restaurants dataset.

**Fig. 17.** Simulated data: the probability the winner is within top-k.
Table 4

Accuracy of DIG for different confidence levels.

<table>
<thead>
<tr>
<th>Confidence level</th>
<th>Simulated data</th>
<th>Netflix data</th>
<th>Sushi data</th>
<th>Pubs data</th>
<th>Restaurants data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>60%</td>
<td>50%</td>
<td>80%</td>
<td>30%</td>
<td>70%</td>
</tr>
<tr>
<td>0.55</td>
<td>60%</td>
<td>50%</td>
<td>80%</td>
<td>30%</td>
<td>90%</td>
</tr>
<tr>
<td>0.6</td>
<td>60%</td>
<td>50%</td>
<td>70%</td>
<td>30%</td>
<td>80%</td>
</tr>
<tr>
<td>0.65</td>
<td>60%</td>
<td>50%</td>
<td>80%</td>
<td>20%</td>
<td>80%</td>
</tr>
<tr>
<td>0.7</td>
<td>80%</td>
<td>50%</td>
<td>70%</td>
<td>40%</td>
<td>80%</td>
</tr>
<tr>
<td>0.75</td>
<td>90%</td>
<td>60%</td>
<td>90%</td>
<td>40%</td>
<td>80%</td>
</tr>
<tr>
<td>0.8</td>
<td>100%</td>
<td>60%</td>
<td>90%</td>
<td>40%</td>
<td>80%</td>
</tr>
<tr>
<td>0.85</td>
<td>100%</td>
<td>60%</td>
<td>90%</td>
<td>40%</td>
<td>80%</td>
</tr>
<tr>
<td>0.9</td>
<td>100%</td>
<td>70%</td>
<td>90%</td>
<td>60%</td>
<td>80%</td>
</tr>
<tr>
<td>0.95</td>
<td>100%</td>
<td>70%</td>
<td>90%</td>
<td>60%</td>
<td>80%</td>
</tr>
</tbody>
</table>

Fig. 18. Netflix data: the probability the winner is within top-k.

Fig. 19. Sushi data - probability winner is within top-k.

Fig. 20. DIG with Majority (MAJ) strategy different skewness levels.

Fig. 21. DIG with Least Misery (LM) strategy different skewness levels.

Fig. 22. ES with Majority (MAJ) strategy different skewness levels.

probabilities are estimations, there is more noise until a 95% probability is reached (Fig. 18). The Sushi dataset also contains probability estimations, but the estimation is more accurate (Fig. 19). To conclude, when the probability estimation is accurate, there is linear relationship between the number of required queries and the approximation level. However, an inaccurate probability distribution results in a “jump” when the required confidence is a 100%.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni–Dunn test concluded that DIG and ES significantly outperform RANDOM at a 95% confidence level.

Another interesting question is whether the confidence level results are accurate. A confidence level refers to the percentage of all possible samples that can be expected to include the true population parameter. The confidence level (1−α) is accurate, if the winner is indeed within the top-k items in (1−α)% of the experiments. We analyzed the accuracy for the DIG algorithm (since it proved to be the best algorithm for approximation settings) for different confidence levels for k = 1. The results are presented in Table 4. As previously shown, since the estimation of the probability distribution of Netflix, Pubs and Restaurants datasets is less accurate, the results for Netflix are less accurate. The accuracy is affected by the bias in the user rating and is beyond the scope of this research. See Koren and Sill (2011) for further details on treating bias.

5.5. Aggregation

We compared the two strategies: Majority (MAJ) and Least Misery (LM) on the DIG (Figs. 20 and 21) and ES heuristics (Figs. 22 and 23) for simulated data with different skewness levels: −6, 0, 6. Axis x presents the required top-k items and axis y presents the
number of queries. DIG and ES with MAJ perform the same for skewness levels 0 and 6, but it is better when the skewness is −6. However, for the DIG and ES with LM, skewness levels have no significant effect on the performance since skewness does not indicate the quantity of low scores in the dataset, and the low scores are exactly the issue that needs to be considered in LM.

A comparison between DIG with MAJ and DIG with LM on simulated data on skewness level 6 (Fig. 24) and on skewness level 0 (Fig. 25) reveals that the LM strategy outperforms MAJ in situations such as these: in a uniform skewness (skewness level 0) and in k > 4 in skewness level −6. This can be explained by the fact that in a setting that is not skewed towards a certain candidate (i.e., any setting apart from −6), there might be more users that voted “1” therefore, LM uses a tie-break to terminate. Thus, LM requires fewer queries in this situation. In the Netflix dataset (Fig. 26) MAJ outperforms LM, further indicating the fact that LM has no additional value when there is no skewness towards a certain winner. Similarly, on the Sushi dataset (Fig. 27), MAJ outperforms LM when k < 5 and then the trend changes and LM outperforms MAJ. On the pubs and restaurant datasets (Figs. 28 and 29) LM outperforms MAJ for both heuristics. These results might be explained by the data skewness.

We compared MAJ and LM with respect to the approximation termination condition, with a constant value of k = 1 on the datasets: Netflix, Sushi, Pubs, and Restaurants (Figs. 30–33). Axis x presents the required confidence level and axis y presents the number of queries. There is no significant difference between MAJ and LM for DIG on the Netflix, Pubs, and Restaurants dataset. For ES, on the other hand, MAJ outperforms LM. This is since ES heuristic does not accommodate any consideration of Least Misery, as it always seeks for the item expected to win, and does not consider the least preferred items. The same results for ES are found on the Sushi dataset (Fig. 31). However, for DIG on the Sushi dataset, LM outperforms MAJ for confidence levels 50%-95%.
confidence level 100%, MAJ outperforms LM. Namely, for one definite winner the system’s entropy can be reduced faster for the Majority aggregation strategy than for the Least Misery strategy probably since Least Misery requires more queries in order to validate that none of the users are miserable.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 90% rejected the null-hypothesis that all heuristics perform the same for different approximation levels. We did not execute the Bonferroni-Dunn test since there is not one algorithm that is preferred over the others.

6. Conclusions

We suggest considering the aggregation strategy and the termination conditions when attempting to reduce preference elicitation communication cost. We examined two termination conditions: selection and approximation. The first condition, selection, returns top-
k items where one of them is the winning item rather than just one (k = 1) definite winning item. The second termination condition, approximation, returns top-
k items with some confidence level \( \alpha (0 \leq \alpha \leq 1) \), rather than top-
k items where one of them is the definite winner (\( \alpha = 1 \)). Furthermore, we examined the Least Misery aggregation strategy and the Majority aggregation strategy.

The final goal of this paper is to employ selection, approximation and aggregation in order to reduce the amount of queries needed during a preference elicitation process for a group of people that want to reach a joint decision. We focused on the Range voting protocol as it is very commonly applied for recommender systems. We implemented two heuristics whose primary aim is to minimize preference elicitation: DIG and ES [Naamani-Dery et al., 2014]. These are the only two publicly available heuristics that aim at reducing preference elicitation for the Range voting protocol. To the best of our knowledge, there are no other algorithms that operate (or can be expanded to operate) under the same settings.

We performed an experimental analysis on two real-world datasets: the Sushi dataset [Kamishima et al., 2005] and the Netflix prize dataset (http://www.netflixprize.com). In order to analyze possible skewness levels in data, we simulated data with different skewness levels. Lastly, we examined real data collected in a user study on a recommender system in Ben Gurion University. We also estimated user-item probability distribution for all datasets.

In general, we show that selecting the suitable aggregation strategy and relaxing the termination condition can reduce communication cost up to 90%. We also show the benefits of the DIG heuristic for reducing the communication cost. In our previous work [Naamani-Dery et al., 2014] we conclude that in most cases the ES heuristic outperforms the DIG heuristic. The ES heuristic focuses on identifying the current local maximum and queries the user that maximizes this item the most. The DIG heuristic focuses on reducing the system entropy. In this paper we reveal that when the termination conditions are relaxed, DIG takes the lead.

We examined how the number of required queries is affected by the request to (1) return one definite winner, and (2) return top-k items. In the latter case, the group members are left with k items to select from (selection termination condition). With respect to the selection condition, there is an inverse linear connection: as k is larger the amount of required queries is reduced. Only when the dataset is skewed towards a certain winner item, and also k is set to \( 0 \leq k \leq 3 \), does ES outperform DIG. This observation assists to determine the conditions in which each of these heuristics should be employed. Also, we can now state that, as expected intuitively, in cases where the group members are willing to accept a set of items rather than one winning item, the communication cost is reduced. For example, if a group’s wish to select a movie can be satisfied with the system offering them a choice of top-3 movies rather than the system determining one movie for them, less queries to group members will be executed.

We studied (1) the tradeoff between finding the optimal winner and thus having an accurate result, and (2) the number of queries required for the process. For the approximation termination condition, we show that the amount of required queries increases proportionally to the confidence level. We show that DIG and ES can output accurate approximate recommendations. However, the accuracy is derived from the dataset’s probability distribution accuracy. When the probability distribution is known or is estimated accurately, the recommendations are more accurate.
With respect to the aggregation strategy, we show that the Majority strategy does not always outperform the Least Misery strategy. It is reasonable to assume that the strategy will be set according to the users’ preferences and not according to the data. We demonstrate the feasibility of choosing either strategy on the datasets.

6.1. Discussion

Our findings append to a growing body of literature on preference elicitation using voting rules (Kalech et al. 2011; Lu & Boutilier 2011). Our research adds a unique contribution to preference elicitation in social choice in a number of perspectives that have previously been overlooked. First, we have studied preference elicitation using a non-ranking protocol (represented by the Range protocol). Previous research has focused only on the ranking Borda protocol. Non-ranking is worth considering since it is abundant and often used by different applications such as netflix.com and booking.com. Secondly, we have suggested methods for reducing the amount of queries: (a) to return a list of top-k items where one of them is the necessary winner; and (b) to approximate the necessary winners or top-k items. These methods offer a decrease in the required amount of queries and have not been previously suggested. Finally, we examined the effect of aggregating the preferences in other strategies but the Majority based strategy. The Least Misery strategy is often needed in real-life scenarios yet has previously been overlooked (e.g., a group looking for a dining location may wish to avoid a fish restaurant if one of the group members dislikes fish).

From the recommender systems domain perspective, this study suggests a framework for preference elicitation that can be used as a second step procedure in group recommenders: to narrow down the predicted items list and present the group of users with definite or approximate necessary winners. Group recommender systems often focus on improving the systems accuracy and usually return a prediction to the group and not definite winning items. A group recommender system can process thousands of candidate items and return a list of top-N items predicted as the most suitable to the group. We can enhance this by eliciting user preferences on these N items and return a definite winner or top-k items (k < N) where one of the items is the winner or an approximate winner with some confidence level. This contribution may add to the usability of a group recommender system offering a platform that enables reaching a joint decision with minimal effort.

As a direct consequence of this study, we encountered a number of limitations, which need to be considered. We assumed that: the user always provides an answer to the query, independence of rating and equal communication cost. This can be overcome by tweaking the model. For example, it is possible to model the probability that the user will answer the query. For a small number of voters and items it is possible to consider dominant probabilities. The communication cost be modeled as a weighted vector and added to the model.

We examined the two aggregation strategies most common in the literature. Extension to other available aggregation strategies does not require a fundamental change since the heuristics and the model do not change. We leave this for future work. Analyzing other termination conditions is yet another promising direction to pursue.

References


