Adaptive Voting Algorithms for Group and Social Recommender Systems

George Popescu

École Polytechnique Fédérale de Lausanne, Switzerland
Systemic Modeling Laboratory (LAMS), BC 167, CH-1015, Lausanne
george.popescu@epfl.ch

Abstract. Nowadays online group activities are emerging, as individuals share their preferences, collaborate, discover and interact with their friends and family. Group recommender systems (GRS) use various social resources to make recommendations of items or activities that users are most likely to consume or agree upon. Thus, aggregating preference and recommending a common set of items for a group has become a challenging topic in online systems providing group suggestions and social websites. This issue is mainly concerned with the following three subjects: eliciting individual users' preferences, suggesting the maximized overall satisfaction outcome for all users and ensuring that the aggregation mechanism is resistant to individual users' manipulation. Furthermore, both individual and group preferences change over time. In order to track all of these changes GRS need to benefit from user interaction. This paper aims to present an innovative algorithm, which adapts to individual preference dynamics for group and social recommender systems. Individuals choose their desired items with the purpose of maximizing the entire group's satisfaction.

1 General Framework

1.1 Notations

A = set of n agents (individuals, users, persons, etc.)

$$a_i \in A, \ \forall 1 \le i \le n \tag{1}$$

S = set of all possible items *k* (alternatives, e.g.: songs)

$$s_j \in S, \ \forall 1 \le j \le m \tag{2}$$

$score(s_j, a_i)$ = score that each agent gives to selected items

$$score(s_j, a_i) \in \{0,1,2,3,4,5\} \ \forall s_j \in l(a_i) \tag{3}$$

$u(s_j, a_i)$ = individual utility and corresponds the score given by each agent

$$u(s_j, a_i) = score(s_j, a_i) \ \forall s_j \in l(a_i), \forall a_i \in A \tag{4}$$

C. Stephanidis (Ed.): Posters, Part II, HCII 2013, CCIS 374, pp. 378–382, 2013.

$l(a_i)$ = list of items of agent a_i

$$l(a_i) = \{s_j \mid score(s_j, a_i) \neq 0\},\ \forall 1 \leq i \leq n \tag{5}$$

$N_i = N_{a_i}$ = set of neighbors of agent a_i, given a trust relationship between agents

$$N_i = \{a_j \mid \exists r \leftrightarrow (a_i, a_j)\},\ \forall 1 \leq i, j \leq n \tag{6}$$

$T_{ij} = T_{a_i a_j}$ = level of **direct** trust between agents a_i and a_j

$$T_{ij} \in N,\ \forall 1 \leq i, j \leq n \tag{7}$$

$\tilde{T}_{ij}$ = level of **indirect** trust between agents a_i and a_j

$$\tilde{T}_{ij} \in N,\ \forall 1 \leq i, j \leq n \tag{8}$$

k = length of the final list of items to be elected

$$k << \sum_{1 \leq i \leq n} l(a_i), \tag{9}$$

1.2 Computation

The motivation of this research is to find a preference elicitation and aggregation method for a group deciding on a common outcome. The method should adapt to the dynamics of the group in terms of preference change, interaction and trust. Two criteria are important for developing it: it must maximize the group satisfaction, and it must encourage users to state their preferences truthfully.

This problem is a general instance of social choice and often modeled as a voting problem. We let A be the set of all agents and S the set of all possible outcomes that can be rated. In a group music recommendation setting, the outcomes are songs s_j to be selected in a common playlist. Each individual a_i submits a numerical vote $score(s_j, a_i)$ for each song s_j. This represents the preference intensity or utility for the respective song.

$$score(s_j, a_i) = \frac{rating(s_j, a_i)}{\sum_j rating(s_j, a_i)} \tag{10}$$

Thus each person estimates self-utility and submits the respective score for each song. Votes are given as ratings, e.g.: 2 out of 5. The voting weight is chosen as a 5-steps Likert scale and represents: 1="Strongly dislike it", 5="Strongly like it". We normalize them so that the scores given by each agent sum to 1. Then, we assign a joint score to each song that is computed as the sum of the scores given by the each individual:

$$score(s_j) = \sum_{a_i \in A} score(s_i, a_j) \tag{11}$$

As such songs' scores are stored and can be listed according in a descending order, for instance. To choose the songs to be included in a playlist of length k, a deterministic method that satisfies montonicity is to choose the k songs with the highest joint rating. This is a generalized plurality rule. However, this method is not truthful.

2 Proposed Voting Algorithm

As a method for choosing a joint playlist, based on the previous discussions, we propose a method we call the probabilistic weighted sum (PWS). PWS is equivalent to the random dictator rule: we iteratively choose each of the k songs randomly:

$$p(s_j) = \frac{score(s_j)}{\sum_{s_j \in S} score(s_j)} \tag{12}$$

The algorithm will choose the playlist by selecting one song after another using this probabilistic distribution. Compared with other social choice based algorithms, PWS is incentive compatible. That is, it is to the best interest of the individual to reveal his/her preferences truthfully. It is in fact equivalent to a random dictator method, where the dictator will choose a song randomly with the probabilities given by its degree of preference – a reasonable method since nobody wants to hear the same song over and over again. The probability of a song s_j to be chosen can be written as:

$$p(s_j) = \frac{score(s_j)}{|A|} = \sum_{a_i \in A} \frac{score(s_j, a_i)}{|A|} \tag{13}$$

In other words, the probability for song s_j to be selected is the probability of choosing user a_i multiplied by the normalized score that user a_i has given to song s_j.

3 Individual Preference Elicitation

The figure below shows the steps followed by each user and the system from rating generation to recommendation. First the users choose their songs. Then they rate those songs corresponding to their utilities. The algorithm normalizes the scores into probabilities as explained above and displays a song list based on these probabilities.

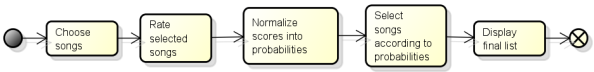

Fig. 1. The steps of the PWS algorithm

The recommendation list of songs favors diversity. Even though users will not get to have their most favorite songs selected, but with a certain probability, the more they interact with the system, the higher the chances will be to their preferences will arrive in the top k. Till this point we have considered only individual interactions with the system, i.e. individual rating. Each user gives personal scores to each of the songs, $score(s_j, a_i)$ which represent numbers from 1 to 5 (ratings).

4 Group Preference Elicitation

We now consider trust between users and define a framework for modeling trust relationships. Suppose users share their songs with other users. If one song is selected by the others then the level of direct trust T_{ij}, between the two users will increase.

For any song s_j we have the following score updating rule:

$$score(s_j) := \sum_{N_j} score(s_j, a_{N_j}) \tag{14}$$

for all neighbors who rated the same song.

$$N_i = \{a_j \mid \exists song \leftrightarrow (a_i, a_j)\}, \forall 1 \le i, j \le n \tag{15}$$

In other words, there exists at least one song rated by both users, not necessary with the same score.

$$N_i = \{a_j \mid l(a_i) \cap l(a_j) \neq \Theta\}, \forall 1 \le i, j \le n \tag{16}$$

N_i is the set of all trusted neighbors for user a_i. Until now we considered only direct trust between users.

Indirect trust can be computed by taking into account the interaction among users. We consider the case in which users critique the preferences of others. Suppose that each user has already created an individual playlist. For each songs in $k(a_i)$ the current user a_i has submitted his/her ratings. Now users can check other users' playlists and can submit their critiques in the form of "like" / "dislike".

$$critique(s_j, a_i) \in \{-1, 0, 1\} \tag{17}$$

$$score(s_j) := \sum_{N_j} score(s_j, a_{N_j}) + \sum_{N'_j} critique(s_j, a_{N'_j}) \tag{18}$$

The total direct trust for user a_i is the sum of all direct trust between all neighboring agents and the current user. We compute this as counting the number of times each other user rated same songs thus adding 1 to the overall direct trust for user a_i.

$$\sum_{j \in N_i} T_{ij} = \sum_{j \in N_j} count(score(s_j, a_i) \geq 3), \ \forall 1 \leq i \leq n \tag{19}$$

Finally, we normalize this direct trust score to 1 allowing a comparison between the levels of direct trust among all users.

$$T_i = T_{a_i} = T(a_i) = \frac{\sum_{j \in N_j} count(score(s_j, a_i) \geq 3)}{\sum_{j \in N_j} count(score(s_j, a_i) \neq 0)}, \ \forall 1 \leq i \leq n \tag{20}$$

We do a similar computation for indirect trust. For both direct and indirect trust we compute $trust(a_i)$ the trustworthiness score to an agent a_i, the sum between the direct and indirect trust for all neighbors.

$$trust(a_i) = T(a_i) + \tilde{T}(a_i), \ \forall 1 \leq i \leq n \tag{21}$$

Users are sorted by their trust coefficient. They are interested in proposing interesting songs and capture the direct and indirect trust of other users. In this way the GRS with the probabilistic weighted sum algorithm increases the group's welfare by giving higher probabilities to songs which "worth" being included in the final top k playlist.

5 Related Work

Addressing the problem of social choice with a general voting mechanism, Brandt and Sandholm (2005) consider the most general case of voting in which the users' rankings of alternatives are mapped to a collective ranking of alternatives by a social welfare functional. Such approach was investigated by: Hastie and Kameda (2005), Herlocker et. al (2004) and Ricci (2009). As an example, CATS is a synchronous collaborative recommender system, which helps a maximum of 4 users to reach a common ski destination using the DiamondTouch tabletop (McCarthy et. al, 2006). TV show experiments revealed important decision making consequences for the impact of scores and recommendations a recommender systems presents to its users (Adomavicius et. al, 2010).