2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops

Group recommendation with automatic identification of users communities

Ludovico Boratto, Salvatore Carta
Dipartimento di Matematica e Informatica
Università di Cagliari
Italy
boratto@sc.unica.it, salvatore@unica.it

Alessandro Chessa
Dipartimento di Fisica
Università di Cagliari
Italy
alessandro.chessa@dsf.unica.it

Maurizio Agelli, M. Laura Clemente
CRS4
Italy
agelli@crs4.it, clem@crs4.it

***Abstract*—Recommender systems usually propose items to single users. However, in some domains like Mobile IPTV or Satellite Systems it might be impossible to generate a program schedule for each user, because of bandwidth limitations.**

A few approaches were proposed to generate group recommendations. However, these approaches take into account that groups of users already exist and no recommender system is able to detect intrinsic users communities.

This paper describes an algorithm that detects groups of users whose preferences are similar and predicts recommendations for such groups. Groups of different granularities are generated through a modularity-based Community Detection algorithm, making it possible for a content provider to explore the trade off between the level of personalization of the recommendations and the number of channels. Experimental results show that the quality of group recommendations increases linearly with the number of groups created.

***Keywords*-recommender systems, collaborative filtering, community detection**

I. Introduction

Recommender systems aim to provide information items (web pages, books, movies, music, etc.) that are likely of interest to a user [10]. Collaborative Filtering (CF) [10] is by far the most successful recommendation technique. The main idea of CF systems is to use the opinions of a community, in order to provide item recommendations. Usually CF systems use these opinions to recommend items to single users.

There are contexts and domains where, however, classic CF systems cannot be used. For example, in multiple access systems with limited transmission capacity like Mobile IPTV or Satellite Systems, it might not be possible to create personalized program schedules for each user. Group recommendations have already been studied from several perspectives. Issues that arise with group modeling have been studied by [7], [5] and in [5] the state-of-the-art in group recommendation is also presented, by doing an overview of the existing techniques, developed for different domains like web/news pages, tourist attractions, music tracks, television programs and movies. These approaches take all for granted the presence of groups of users with similar opinions. However, usually there's no a priori knowledge about the users partition in groups and, in such cases, the problem relies in identifying groups of related users. Automatic detection of communities in group recommenders is not a trivial issue, since communities have to be found just considering users opinions. Moreover, meaningful informations about groups preferences have to be retrieved and exploited, in order to provide good recommendations.

In this paper we propose an algorithm to generate group recommendations, able to detect intrinsic communities of users whose preferences are similar. The algorithm takes as input a matrix that associates a set of *users* to a set of *items* through a *rating*. We'll call this matrix the *ratings matrix*. Based on ratings expressed by each user in the ratings matrix, our algorithm evaluates the level of similarity between users and generates a network that contains the similarities. A modularity-based Community Detection algorithm proposed by [1] will be run on the network, in order to find partitions of users in communities. For each community, ratings for all the items will be calculated.

Since the Community Detection algorithm is able to produce a dendrogram, i.e. a tree that contains hierarchical partitions of the users in communities of increasing granularity, experiments were conducted in order to evaluate the quality of the recommendation for the different partitions. Results show that the quality of group recommendations increases linearly with the number of communities created.

The scientific contribution of the recommendation algorithm proposed is the capability to automatically detect intrinsic communities of users who share similar preferences, making it possible for a content provider to explore the trade off between the level of personalization of the recommendation and the number of channels.

The rest of the paper is organized in the following way: section II contains a detailed description of the steps we followed to build our algorithm; section III describes the experiments we conducted and outlines main results; section IV contains comments and future developments.

II. Group recommendation with automatic identification of users communities

A. Top level view of the algorithm

The proposed group recommendation algorithm works in four steps:

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.346

1) Users similarity evaluation: In order to create communities of users, the algorithm takes as input a *ratings matrix* and evaluates through a standard metric (cosine similarity) how similar the preferences of two users are. The result is a weighted network where nodes represent users and each weighted edge represents the similarity value of the users it connects. A post-processing technique is then introduced to remove noise from the network and reduce its complexity.

2) Communities detection: To identify intrinsic communities of users, a Community Detection algorithm proposed by [1] is applied to the users similarity network and partitions of different granularities are generated.

3) Ratings prediction for items rated by enough users of a group: A group's ratings are evaluated by calculating, for each item, the mean of the ratings expressed by the users of the group. In order to predict meaningful ratings, our algorithm calculates a rating only if an item was evaluated by a minimum percentage of users in the group. With this step it is not possible to predict a rating for each item, so another step has been created to predict the remaining ratings.

4) Ratings prediction for the remaining items: For some of the items, ratings could not be calculated by the previous step. In order to estimate such ratings, similarity between items is evaluated, and the rating of an item is predicted considering the items most similar to it.

The next sections will describe in detail the four steps that constitute the algorithm.

B. Step 1. Users similarity evaluation

Here we describe how a ratings matrix can be used to evaluate the strength of the similarity between users.

Let v_i be the vector of the ratings expressed by a user i for the items and v_j be the vector of the ratings expressed by a user j for the items. The similarity s_{ij} between user i and user j can be measured by the cosine similarity between the vectors:

$$s_{ij} = cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \times \|v_j\|}$$

The similarities can be represented in a network, the *users similarity network*, that links each couple of associated users with a weighted edge.

As highlighted by [4], in networks like the one we built, edges have intrinsic weights and no information is given about the real associations between the nodes. Edges are usually affected by noise, which leads to ambiguities in the communities detection. Moreover, the weights of the edges in our network are calculated considering the ratings and it is well known that people have different rating tendencies: some users tend to express their opinion using just the end of the scales, expressing if they loved or hated an item. In order to eliminate noise from the network and reduce its complexity by removing weak edges, a parameter called *noise* was set in our algorithm. The parameter will indicate the weight that will be subtracted by every edge.

C. Step 2. Communities Detection

This step of our algorithm has the goal to find intrinsic communities of users, accepting as input the weighted users similarity network that was built in the previous step. Another requirement is to produce the intrinsic users communities in a hierarchical structure, in order to deeper understand and exploit its inner partition. Out of all the existing classes of clustering algorithms, we identified the class the of complex network analysis [3] as the only class of algorithms fulfilling our requirements. In 2004 a new optimization function has been introduced, the modularity [8], that measures for a generic partition of the set of nodes in the network, the number of internal (in each partition) edges respect to the random case. The optimization of this function gives, without a previous assessment of the number and size of the partitions [3], the natural community structure of the network. Moreover it is not necessary to embed the network in a metric space like in the k-means algorithm. A notion of distance or link weight can be introduced but in a pure topological fashion [9].

Recently a very efficient algorithm has been proposed, based on the optimization of the weighted modularity, that is able to easily handle networks with millions of nodes, generating also a dendrogram; a community structure at various network resolutions [1]. Since the algorithm had all the characteristics we were looking for, it was chosen to create the groups of users used by our group recommendation algorithm.

D. Step 3. Ratings prediction for items rated by enough users of a group

In order to express a group's preference for an item, our algorithm calculates its rating, considering the ratings expressed by the users of the community for that item.

An average is a single value that is meant to typify a list of values. The most common method to calculate such a value is the arithmetic mean, which also seems an effective way to put together all the ratings expressed by the users in a group. So, for each item i, its rating r_i is expressed as:

$$r_i = \frac{1}{n} \sum_{u=0}^{n} r_u$$

where n is the number of users of the group who expressed a rating for item i and r_u is the rating expressed by each user for that item. In order to calculate meaningful ratings for a group, a rating r_i is considered only if a consistent part of the group has rated the item. This is done through a parameter, called *co-ratings* which expresses the minimum percentage of users who have to rate an item in order to calculate the rating for the group.

E. Step 4. Ratings prediction for the remaining items

For some of the items, ratings could not be calculated by the previous step. In order to estimate such ratings, we

built a network that contains similarities between items. Like the users similarity network presented in II-B, the network is built through the ratings matrix, considering the ratings expressed for each item. Let w_i be the vector of the ratings expressed by all the users for item i and w_j be the vector of the ratings expressed by all the users for item j. The similarity t_{ij} between item i and item j is measured with the cosine similarity and the similarities are represented in a network called *items similarity network*, from which noise was removed through the *noise* parameter presented in II-B.

For each item not rated by the group, a list is produced with its nearest neighbors, i.e. the most similar items already rated by the group, considering the similarities available in the *items similarity network*. Out of this list, the *top* items are selected. Parameter *top* indicates how many similarities the algorithm considers to predict the ratings.

An example of how the *top* similar items are selected is shown in Figure 1. The algorithm needs to predict a rating for Item 1. The most similar items are shown in the list. For each similar item j, the table indicates the similarity with Item 1 (column t_{1j}) and the rating expressed by the group (column r_j). In the example, the *top* parameter is set to 3 and items with similarity 0.95, 0.88 and 0.71 are selected.

Item j	t_{1j}	r_j
Item 2	0.95	3.5
Item 3	0.95	4.2
Item 4	0.88	2.8
Item 5	0.71	2.6
Item 6	0.71	3.9
Item 7	0.71	4.3
Item 8	0.63	1.2
Item 9	0.55	3.2

Figure 1. Top similar items of an unrated item

We can now predict the rating of an unrated item by considering both the rating and the similarity of its *top* similar items:

$$\bar{r}_i = \frac{\sum_{j=0}^{n} r_j \cdot t_{ij}}{\sum_{j=0}^{n} t_{ij}}$$

where n is the number of items selected in the list. Given the example in Figure 1, $\bar{r}_1 = 3.55$.

To make meaningful predictions, we need evaluate how "reliable" our predictions are. This is done by calculating the mean of the *top* similarities and by setting a *trust* parameter. The parameter indicates the minimum value the mean of the similarities has to get, in order to be considered reliable and consider the predicted rating. The mean of the similarities in the previous example is 0.85 so, to consider $\bar{r}_1$, the *trust* parameter has to be lower than 0.85.

III. Algorithm Experimentation

In order to evaluate the quality of the recommendations, our algorithm was tested using MovieLens[1], a dataset widely used to evaluate CF algorithms. We built a framework that extracts a subset of ratings from the dataset, predicts group recommendations through the proposed algorithm and measures the quality of the predictions in terms of RMSE. The rest of this section will describe the details of the algorithm experimentation.

[1]http://www.grouplens.org/

A. Experimental methodology and setup

The experimentation was made through the MovieLens dataset, which is composed of 1 million ratings, expressed by 6040 users for 3900 movies. To evaluate the quality of the ratings predicted by our algorithm, around 10% of the ratings was extracted as a probe test set and the rest of the dataset was used as a training set for the algorithm.

The group recommendation algorithm was run with the training set and, for each partition of the users in communities, ratings were predicted. The quality of the predicted ratings was measured through the Root Mean Squared Error (RMSE). The metric compares the probe test set with the ratings predicted: each rating r_i expressed by a user u for an item i is compared with the rating $\bar{r}_i$ predicted for the item i for the group in which user u is. The formula is shown below:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(r_i - \bar{r}_i)^2}{n}}$$

where n is the number of ratings available in the test set. To evaluate the performances of the proposed algorithm, we compared them with the results obtained considering a single group with all the users (predictions are calculated considering all the preferences expressed for an item), and the results obtained using a classic CF algorithm proposed in [2], where recommendations are produced for each user.

B. Experimental results

To evaluate our algorithm's performances we studied the quality of the recommendations, considering different values of each parameter. The only value that could not be changed was *noise*, because if we subtracted more than 0.1 to the edges of the *users similarities network*, the network would become disconnected.

The first experiment conducted was to evaluate the quality of the recommendations for different values of the *co-ratings* parameter, i.e. the minimum percentage of users who have to rate an item, in order to calculate the rating for the group. Parameter *top* was set to 2 and parameter *trust* was set to 0.0. Figure 2 shows how RMSE varies with the number of groups, for different values of *co-ratings* (10%, 20% and 25%). We can see that as the number of groups grows, the quality of the recommendations improves, since groups get smaller and our algorithm predicts more precise ratings. To conduct the following experiments, the value of *co-ratings* chosen was 20%. The next experiment we conducted was to evaluate the quality of recommendations for different values of the *top* parameter, i.e. the number of similarities

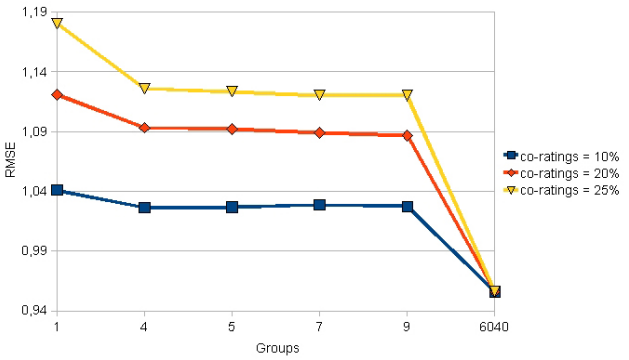

Figure 2. Algorithm's performances with different co-ratings values

considered to select the nearest neighbors of an item. We won't present the plot in this paper, since the results show that the quality of recommendations doesn't depend from this parameter and RMSE doesn't change. The initial value of 2 was kept to conduct the next experiment.

The last parameter to evaluate is *trust*, i.e. the minimum value the mean of the similarities has to get when the algorithms predicts a rating considering the nearest neighbors of an item. Figure 3 shows how RMSE varies with the number of groups, for different values of the parameter (0.0, 0.1 and 0.2). In Figure 3 is shown that the quality of the

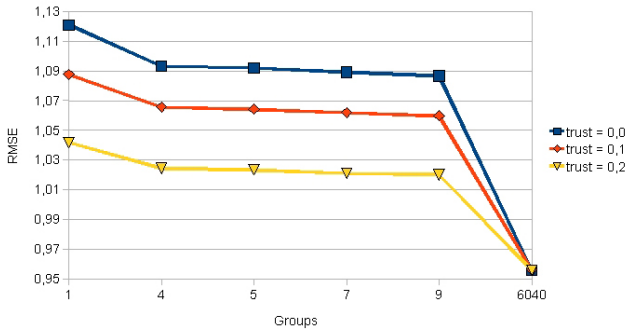

Figure 3. Algorithm's performances with different trust values

performances improves for higher values of *trust*, i.e. when the ratings predicted can be considered more "reliable".

IV. Conclusions and Future Work

In this paper we proposed a group recommendation algorithm that, based on users' preferences, detects communities of similar users and predicts groups preferences. Experimental results show that the quality of the recommendations generated by our algorithm improves linearly with the number of communities created. The proposed technique will be improved in several ways: new metrics will be used to evaluate similarities between users and similarities between items, the structure of communities will be studied to improve the effectiveness of a group's ratings and we'll test our algorithm on different datasets to evaluate its performances.

Acknowledgment

The authors would like to thank Marco Gaviano, Giorgio Porcu and Luca Urru for participating in the definition and implementation of the proposed algorithm.

References

[1] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008+, October 2008.

[2] Maria Laura Clemente. Experimental results on item-based algorithms for independent domain collaborative filtering. In *AXMEDIS '08: Proceedings of the 2008 International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution*, pages 87–92, Washington, DC, USA, 2008. IEEE Computer Society.

[3] Santo Fortunato and Claudio Castellano. Community structure in graphs. *Springer's Encyclopedia of Complexity and System Science*, Dec 2007.

[4] D. Gfeller, J. C. Chappelier, and De Los. Finding instabilities in the community structure of complex networks. *Physical Review E*, 72(5 Pt 2):056135+, November 2005.

[5] Anthony Jameson and Barry Smyth. Recommendation to groups. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer, 2007.

[6] H. Lieberman. Let's browse: a collaborative browsing agent. *Knowledge-Based Systems*, 12(8):427–431, December 1999.

[7] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85, 2004.

[8] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 69(2 Pt 2), February 2004.

[9] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004.

[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[11] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.