Improving Sparsity Problem in Group Recommendation

Sarik Ghazarian[‡], Nafiseh Shabib[‡†], Mohammad Ali Nematbakhsh[‡]
[‡]University of Isfahan, [‡†]Norwegian University of Science and Technology
sarikghazarian@yahoo.com,
shabib@idi.ntnu.no,
mnematbakhsh@eng.ui.ir

ABSTRACT

Group recommendation systems can be very challenging when the datasets are sparse and there are not many available ratings for items. In this paper, by enhancing basic memory-based techniques we resolve the data sparsity problem for users in the group. The results have shown that by conducting our techniques for the users in the group we have a higher group satisfaction and lower group dissatisfaction.

Keywords

sparsity, group recommendation, collaborative filtering

1. INTRODUCTION

Recommendation systems (RSs) are tools and techniques, which provide suggestions for items to be used by users. They generally directed towards helping users for finding items that are likely interested in the overwhelming number of items and they try to predict the most suitable products or services, based on the users' preferences and constraints. However, even active users have rated just a few items of the total number of available items in a database and respectively, even popular items have been rated by only a few number of total available users in the database. This problem, commonly referred as a sparsity problem [17]. Different approaches have been proposed in the research literature focusing on Sparsity problem for single user recommendations [21, 24]. However, as far as we know, this is the first work presenting a complete model for group recommendations, which resolving sparsity problem for a group. In general, sparsity has a major negative impact on the effectiveness of a collaborative filtering approach and especially on group recommendation. The main challenge behind group scenarios has been that of computing recommendations from a potentially diverse set of group members' ratings in a sparse situations. In this work, we studied sparsity problem in the group recommendation. First, we formalize the problem of *sparsity* in the group recommendation and use our model for aggregating user rating in a group. Second, we run an extensive set of experiments with different group sizes and different group cohesiveness on Millions of Song data set. Our experiments exhibit that in the most cases the group satisfaction in our proposed model is higher and the group dissatisfaction is lower than the previous models, which does not take into account sparsity.

The rest of paper is organized as follows: Section 2 describes the sparsity problem for a group and we propose a complete model for sparsity in the group recommendation. Experiments are presented in section 3. Section 4 provides some background and formalism. We conclude in section 5.

2. DATA MODEL AND RECOMMENDATION ALGORITHM

We assume a set of users $\mathcal{U} = \{u_1, \ldots, u_n\}$ out of which any ad-hoc group $\mathcal{G} \subseteq \mathcal{U}$ can be built. We consider a set $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ with m items.

2.1 Item-Item Similarity

The basic component of proposed method is a machine learning regression method called Support Vector Machine (SVM) which is used for calculating similarities between items [26]. SVM is a supervised learning technique, which learns the function that is produced from input data in the best manner. It uses the built-in function to give appropriate output for an input data [26]. The input data pairs are as follows: $(x_1, y_1), ..., (x_i, y_i)$. The x_i is a record in d dimensional space and y_i is a real value. SVM tries to find $f(x)$ function which approximates the relations between data points [20]. The target function has two types: *linear* and *nonlinear*. In linear regression the relationships between input and output data points are linear and their relationships can be approximated by a straight line. The linear function is computed as equation 1.

$$f(x) = w.x + b \tag{1}$$

, where $w \in X$, X is the input space and b is a real value [20].

In nonlinear case SVM preprocesses input data. It uses nonlinear mapping function ($\varphi \rightarrow \rho$) which maps data from input space to the new feature space ρ. After this mapping action, the standard linear SVM regression algorithm is applied in the new higher feature space. The dot product between data points in higher dimensional feature is called kernel function [23]. Equation 2 shows this function.

$$K(x, x') = \varphi(x).\varphi(x') \tag{2}$$

There are different kernel functions like linear, polynomial, radial basis function (RBF), and Pearson VII Universal Kernel (PUK) [23]. In our proposed method PUK function has been used for modeling the similarities between items, because it had higher accuracy than other functions.

$$PUK : k(x, x') = \frac{1}{\left[1 + \left(\frac{\sqrt[2]{\|x-x'\|^2}\ \sqrt[2]{2^{(\frac{1}{\omega})}-1}}{\sigma}\right)^2\right]^{\omega}} \tag{3}$$

2.2 Listen Count

The algorithms in our work are based on explicit feedback from users; subsequently there is a need to normalize the listening counts to a predefined scale so that the algorithms can work optimally. In the [11], they modified basic latent factor model to convert implicit ratings to the explicit ones. Similarly to the approach taken [11], a boolean variable (p_{ui}) shows the user's interest on an item (equation 4). If a user has listened to a song (l_{ui}), its boolean variable's value is 1 otherwise it is 0. Thus, implicit data do not indicate users' preferences, rather they show confidence (c_{ui}) about users' preferences and there is a direct relationship between confidence value and the number of times that each user has listened to a song (equation 5). The relationship is controlled by constant α.

$$p_{ui} = \begin{cases} 1 & \text{if } l_{ui} > 0 \\ 0 & \text{if } l_{ui} = 0 \end{cases} \tag{4}$$

$$c_{ui} = 1 + \alpha l_{ui} \tag{5}$$

By these alternations, the equation of latent factor model modified as equation 6. This equation is a least square optimization process by considering user factors (p_u) or item factors (q_i) to be fix in each step. After finding user factors and item factors, their dot products show the users' explicit ratings on items.

$$\min_{q^*,p^*} = \sum_{r_{ui} \text{is known}} c_{ui}(r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{6}$$

2.3 Sparsity Calculation

The sparsity value was computed as follows:
The ratio of specified ratings of items in the initial user-item matrix to the whole specified and not specified items' ratings.

$$SparsityValue = \frac{\text{Num.of specified ratings}}{\text{Num.of all possible ratings}} \tag{7}$$

2.4 Group Modeling

We define the following hypothesis: *The relevance between a group and an item i is only dependent on the relevance of i to individual members of the group.* Using this hypothesis, we derive the following definition that not only includes the preferences of individual users but also integrates the users preferences when they are in a group while recommending a set of items.

2.4.1 User-User Similarity

The major goal of this component is to overcome the weakness of Pearson's correlation method in the sparsity situation. The Pearson's correlation is limited to the joint items in both users' preference lists. In a random group setting, the collections of common items between users are very small, so comparing users based on very few items leads to lower accuracy [8, 19]. To solving this problem, the idea of proposed method is to compare all items rated by one user with all items in another user in the group, one by one. In other words, our method involves all possible combination of items in preference lists of both users. Equation 8 demonstrates the idea. The basic part of this equation is based on our conception of similar and dissimilar users:
Two users are considered similar, if they have close ratings for similar items.
Two users are dissimilar, if they have rated two dissimilar items.
Given a group $\mathcal{G}$, the similarity of each user $u \in \mathcal{G}$ is denoted as:

$$UserSim_{uv} = \frac{\Sigma_{\forall i \in R_u \bigcap \forall j \in R_v}(1 - \frac{|r_{ui} - r_{vj}|}{r_{max} - r_{min}}) \times ItemSim_{ij}}{\Sigma_{\forall i \in R_u \bigcap \forall j \in R_v}|ItemSim_{ij}|} \tag{8}$$

, $R_u = \{i | r_{ui} \neq 0\}, R_v = \{j | r_{vj} \neq 0\}$, and r_{max} and r_{min} are maximum and minimum possible values of the ratings. Note that, $ItemSim_{ij}$ is equal to similarity values between items i and j which is calculated by the SVM regression model that has been explained in the 2.1.

2.4.2 User-Item Relevance

Given a group $\mathcal{G}$, the relevance of a user $u \in \mathcal{G}$ for an item $i \in \mathcal{I}$ is denoted as:

$$Rel_{ui} = \bar{r}_u + \frac{\Sigma_{v \in U}(r_{vi'} - \bar{r}_v) \times UserSim_{uv}}{\Sigma_{v \in U}|UserSim_{uv}|} \tag{9}$$

, where i' is the most similar item to i that user v has rated. Thus, by considering i' in the relevance function, it is not required to take into account just the users who have rated the same item, but it considers all ratings given by users, and we can use ratings of other most similar items to the target item to fill in the sparseness.

2.4.3 Group Relevance

The preference of an item i by a group $\mathcal{G}$, denoted as $Grel(\mathcal{G}, i)$, is an aggregation over the preferences of each group member for that item. We consider two main aggregation strategies:
Average

$$Grel(\mathcal{G}, i) = \frac{\Sigma_{u \in \mathcal{G}} Rel_{ui}}{|\mathcal{G}|} \tag{10}$$

Least Misery

$$Grel(\mathcal{G}, i) = min_{u \in \mathcal{G}}(Rel_{ui}) \tag{11}$$

2.5 Group Satisfaction

To evaluate our methods accuracy in group recommendation process, we used group satisfaction metric [5].
This metric is the average of all group members' satisfaction for recommended items

$$Gsat = \frac{\Sigma_{u \in U} Usat}{|\mathcal{G}|} \tag{12}$$

User's satisfaction is shown as $Usat(u)$ which is calculated:

$$Usat = \frac{\Sigma_{i=1}^{k} Rel_{ui}}{k * Max(Rel_{ui})} \tag{13}$$

, where Rel_{ui} is user preference on item, k is the number of items, and $Max(Rel_{ui})$ is maximum preference value of user u for all items.

2.6 Group DisSatisfaction

To evaluate our methods in group recommendation process, we also used group dissatisfaction metric [13]. This metric is the fraction of dissatisfied users whose satisfaction measures were less than a threshold. In our case we consider

the threshold equals to 0.6.

$$GdisSat = \frac{|\mathcal{U}|}{|\mathcal{G}|} \quad (14)$$

, where $u|Usat < 0.6$ (equation 13)

3. EXPERIMENTS

We have shown after solving sparsity problem for each single user in the group, we have a higher group satisfaction and lower group dissatisfaction.

Dataset description: In this section, we evaluate our method with Million Song Dataset (MSD)[1], in the music recommendation scope. The Million Song Dataset (MSD) is a collection of music audio features and metadata that has created to support research into industrial-scale music information retrieval. It is freely-available collection of meta data for one million of contemporary songs such as song title, artist, publication year, audio features, and much more [14]. In addition, The MSD is a cluster of complementary datasets contributed by the community: SecondHandSongs dataset for cover songs, musiXmatch dataset for lyrics, Last.fm dataset[2] for song-level tags and similarity, and Taste Profile subset for user listening history data. Comprising several complementary datasets that are linked to the same set of songs, the MSD contains extensive meta-data, audio features, song-level tags, lyrics, cover songs, similar artists, and similar songs. In this work, we have used information about song's features such as *title*, *release*, *artist*, *duration*, *year*, *song-hotness*, *songs similarity*, *users listening history*, and *song's tags*. In addition to this information, we have information about song tags and its degrees in Last.fm dataset, which the tag's degree shows how much the song is associated to a particular tag. In our work, for each song we consider three main tags.

We implemented our prototype system using Java and for computing SVM model's accuracy we used WEKA[3].

3.1 Item-Item Similarity

In order to use similarity data between songs and create SVM regression model, we needed to prepare suitable data, *preprocessing*, for training process as follows: *song*, *release*, *artist*, *term1*, *term2*, *term3*, *song-hotness*, *duration*, *year*, *similarity-degree*.

In MSD, about half of songs have at least one tag. In this research for each song, its three most relevant tags were considered. If a song didn't have three relevant tags, remaining tags were filled with the highest one. Similarity-degree is an integer attribute in [0, 1] interval. 1 shows the most similar songs and conversely 0 is used for dissimilar songs. In SVM model each record should be represented as a point in input space. To achieve this purpose similarity based functions have been used [10]. For computing similarity between string attributes, *Jaro-Winkler* method has been used, which gives 1 to most similar items and 0 to dissimilar ones. For terms, we used similarity function of nominal attributes. After computing similarity between corresponding pairs of attributes, each record came in form: *title-dif*, *release-dif*, *artist-dif*, *term-dif*, *song-hotness-dif*, *duration-dif*, *year-dif*, *similarity-degree* The "dif" suffix stands for the differences. Then we used these new records to create SVM model for predicting similarities between songs.

3.1.1 Item-Item similarity results

For computing SVM model's accuracy, mean absolute error (MAE) [25] values of different regression models were compared by using Waikato Environment for Knowledge Analysis (WEKA) software tool. All parameters in different methods were tested. In all SVM methods with different kernel functions like PUK, RBF, $normalizedPolyKernel$, and $polyKernel$, the PUK kernel function with $\sigma = 1$ and $\omega = 1$ had the minimum and best MAE value. Figure 1 illustrates different MAE values for different regression methods in WEKA. Therefore, in our work PUK function has been used for modeling the similarities between items.

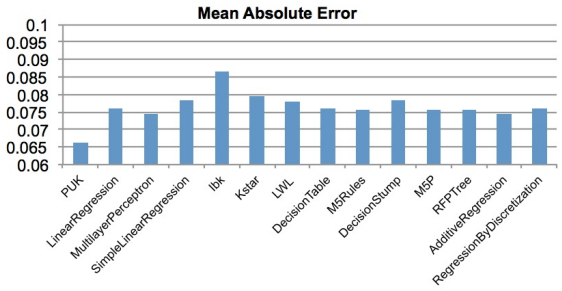

Figure 1: MAE value for different regression methods

3.2 User Collection Phase

We selected subset of users to provide their music preferences. Later, those users are used to form different groups and perform judgments on group recommendations. For this aim, we selected those users who have at least listened to fifteen songs in our dataset. As mentioned in previous section, MSD contains listening history of users, which shows the number of times each user has listened to a particular song. Thus, preferences have been expressed in implicit format. This format is not equivalent to explicit one, which shows the exact preferences of users. Since, the user-based and item-based collaborative filtering (CF) approaches have been designed for explicit ratings, conversion of implicit feedbacks to explicit ones was essential. In order to achieve explicit one, we have used latent factor model with some alternations as proposed in the Listen Count in the previous part.

3.3 Group Formation

We considered two main factors in forming user groups *i.e.* *group size*, *group cohesiveness* [2]. We hypothesized that varying group sizes will impact to the group satisfaction. We chose three group sizes, 3, 5, and 10, representing small, medium, and large groups, respectively. Similarly, we assumed that group cohesiveness (i.e., how similar are group members in their music tastes) is also a significant factor in their satisfaction with the group recommendation. As a result, we chose to form three kinds of groups: *similar*, *dissimilar*, and *random*.

3.4 Result Interpretation

After predicting unknown items' score in all users' preference lists, it is essential to aggregate users' preferences to

[1]http://labrosa.ee.columbia.edu/millionsong
[2]http://last.fm
[3]http://www.cs.waikato.ac.nz/ml/weka

make recommendation for a group. For this purpose, we used basic methods (average and least misery) and recommended k items with highest values. To evaluate our method in the group recommendation process, we used group satisfaction and dissatisfaction metrics. The reason that we used group dissatisfaction metric is observing how the algorithm performs when we have dissatisfied members in the group. Note that, the sparsity value for each group is the following numbers.
Similar group : G3=0.31 G5=0.55 G10=0.77
Dissimilar group : G3=0.52 G5=0.68 G10=0.80
Random group : G3=0.58 G5=0.72 G10=0.84

3.4.1 Varying Group size

We examined the effect of different group sizes on group satisfaction/dissatisfaction in Figure 2. The number of recommended items is fixed 10 and the group sizes varies between 3, 5, and 10 members. As we can see in Figure 2, in the similar groups, the group satisfaction remains the same even though the number of people in each group is increasing. In addition, in most of cases our algorithm has higher group satisfaction in both average and least misery methods in compare of CF method, which does not take into account sparsity. Additionally, with increasing the group sizes the sparsity value is increasing, but our algorithm performs fairly constant. Moreover, the result shows that in the dissimilar and random groups we have lower dissatisfaction.

3.4.2 Varying Top-k

We examined the effect of different recommendation items (Top-k= 5,10,15, and 20) on group satisfaction/dissatisfaction in Figure 3. The group size is fixed 10. The result shows that with increasing the number of items, the group satisfaction is decreasing in all the groups but it decreases more in the similar and dissimilar groups than random groups. In general, our method has a higher group satisfaction in compare of CF method. Also, the result shows that, we have less dissatisfaction when we applied Average as an aggregation method and we have less dissatisfaction in our method.

3.4.3 Varying Group Cohesiveness

We examined the effect of different group cohesiveness on group satisfaction/dissatisfaction in Figure 4. Group cohesiveness varies between similar group (similarity between members >0.5), dissimilar (similarity between members <0.5) and random members. The number of recommended item is fixed 10. Our observation showed that for small groups, group satisfaction is very close to each other in different techniques, but in the random groups we can see noticeably change in the group satisfaction between CF and our proposed method that takes into account sparsity. In addition, the result shows that in the dissimilar and random group our method has a lower dissatisfaction.

4. RELATED WORK

Research on recommendations is extensive. Typically, recommendation approaches are distinguished between: content-based, collaborative filtering, and hybrid [1]. Recently, there are also approaches focusing on group recommendations. Group recommendation aims to identify items that are suitable for the whole group instead of individual group members. Group recommendation has been designed for various domains such as news pages [18], tourism [9], music [6], and TV programs [27]. Group is defined as two or more individuals who are connected to one another. A group can range in size from two members to thousands of members. A group may be formed at any time by a random number of people with different interests, a number of persons who explicitly choose to be part of a group, or by computing similarities between users with respect to some similarity functions and then cluster similar users together [15, 2]. There are two dominant strategies for groups: (1) aggregation of individual preferences into a single recommendation list or (2) aggregation of individual recommendation lists to the group recommendation list [2, 3]. In other words, the first one creates a pseudo user for a group based on its group members and then makes recommendations based on the pseudo user, while the second strategy computes a recommendation list for each single user in the group and then combines the results into the group recommendation list.
However, in the both approaches we may faced the sparsity problem. Sparsity is one of the major problems in memory-based CF approaches [22]. In sparseness conditions most cells of user-item matrix are not rated. The reason is that users may not willing to provide their opinions and preferences and they do this only when it is necessary [7]. In these type of matrices, the accuracy of calculated predictions by applying memory-based CF approaches is low, since there are not enough information about user ratings [12]. Lately, Ntoutsi applied user-based CF approach in order to predict unknown ratings [16]. For this, they partitioned users in to clusters. Then for predicting a particular item's rating for a user, they considered just the ones in the cluster of target user instead of all users in dataset. They calculated the relevancy of an item to a user based on the relevancy of that item to similar users in the target user's cluster. Moreover, they involved a support score in prediction process to be shown how many users in the cluster have rated that item. Because of using memory-based approaches as basis, this approach also cannot be used in sparse data situations. Chen et al. proposed a method which predicts each item's group rating by considering its similar items that have been rated by whole group or by most subgroups [4]. For this aim, first they applied collaborative filtering technique and find each user's preferences on that item and then used genetic algorithm according to subgroups' ratings to achieve the item's overall score. However, our main focus in this research is on sparsity problem in users' preference lists, Chen et al. worked on sparsity problem in groups' ratings, for this reason they could use collaborative filtering in their calculations.

5. CONCLUSION

We formalize the problem of sparsity in the group recommendation and use our model for aggregating user rating for the group. In this work, we proposed a new method that overcomes the weakness of basic memory-based approaches in sparsity. We evaluated our method in sparse cases and compared it with prior methods. The results show that in sparse matrices our proposed method has better group satisfaction and lower group dissatisfaction than basic CF. In addition, in conditions where user-based approach can be run, our proposed method performs better. In the future, we plan to peruse the accuracy of our proposed method in other less been paid fields like TV programs, books and images, and we want to investigate our research in the big

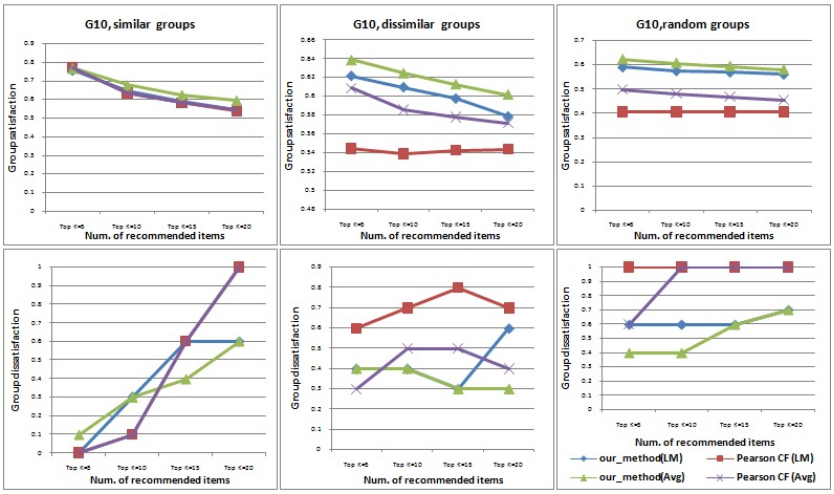

Figure 2: Comparison of group satisfaction and group dissatisfaction with varying group size

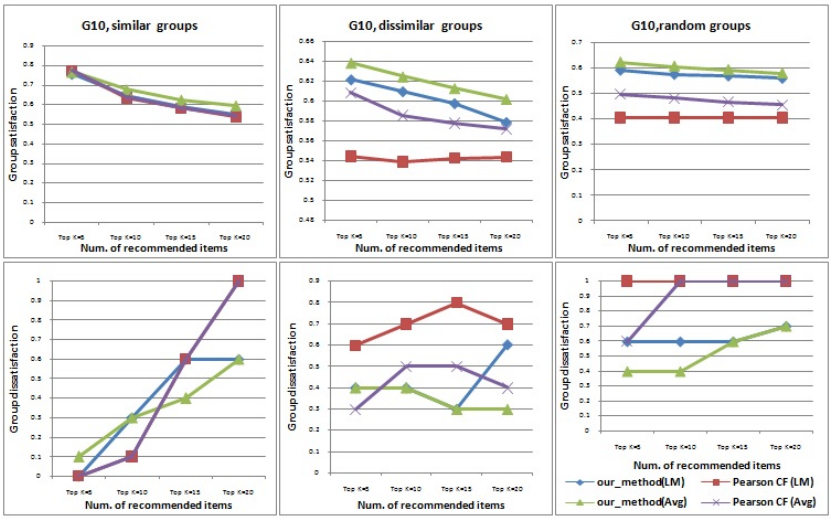

Figure 3: Comparison of group satisfaction and group dissatisfaction with varying Top-k

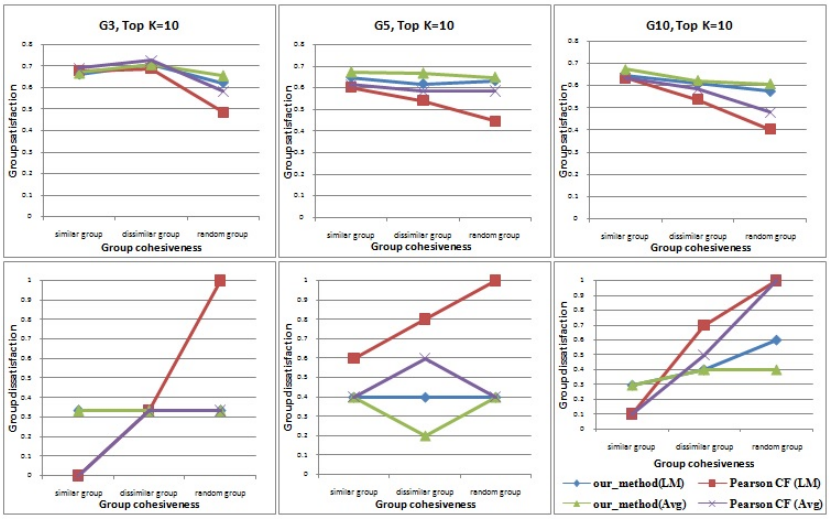

Figure 4: Comparison of group satisfaction and group dissatisfaction with varying group cohesiveness

groups.

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] S. Amer-Yahia, S. B. Roy, A. Chawlat, G. Das, and C. Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, pages 754–765, 2009.

[3] S. Berkovsky and J. Freyne. Group-based recipe recommendations: Analysis of data aggregation strategies. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 111–118, 2010.

[4] Y.-L. Chen, L.-C. Cheng, and C.-N. Chuang. A group recommendation system with consideration of interactions among group members. *Expert Syst. Appl.*, 34(3):2082–2090, 2008.

[5] I. A. Christensen and S. N. Schiaffino. Entertainment recommender systems for group of users. *Expert Syst. Appl.*, 38(11):14127–14135, 2011.

[6] A. Crossen, J. Budzik, and K. J. Hammond. Flytrap: Intelligent group music recommendation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 184–185, New York, NY, USA, 2002. ACM.

[7] L. N. Dery. Iterative voting under uncertainty for group recommender systems (research abstract). In J. Hoffmann and B. Selman, editors, *AAAI*. AAAI Press, 2012.

[8] A. Eckhardt. Similarity of users' (content-based) preference models for collaborative filtering in few ratings scenario. *Expert Syst. Appl.*, 39(14):11511–11516, Oct. 2012.

[9] I. Garcia, L. Sebastia, and E. Onaindia. On the design of individual and group recommender systems for tourism. *Expert Syst. Appl.*, 38(6):7683–7692, June 2011.

[10] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[11] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE.

[12] Z. Huang, H. Chen, and D. D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

[13] J. K. Kim, H. K. Kim, H. Y. Oh, and Y. U. Ryu. A group recommendation system for online communities. *Int. J. Inf. Manag.*, 30(3):212–219, June 2010.

[14] B. McFee, T. Bertin-Mahieux, D. P. Ellis, and G. R. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 909–916. ACM, 2012.

[15] E. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. Fast group recommendations by applying user clustering. In *Proceedings of the 31st International Conference on Conceptual Modeling*, pages 126–140. Springer-Verlag, 2012.

[16] E. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. Fast group recommendations by applying user clustering. In *ER*, pages 126–140, 2012.

[17] M. Papagelis, D. Plexousakis, and T. Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In *Proceedings of the Third International Conference on Trust Management*, iTrust'05, pages 224–239, Berlin, Heidelberg, 2005. Springer-Verlag.

[18] S. Pizzutilo, B. De Carolis, G. Cozzolongo, and F. Ambruoso. Group modeling in a public space: Methods, techniques, experiences. In *Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, AIC'05, pages 175–180, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS).

[19] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender systems: A case study. In *WebKDD Workshop at the ACM SIGKKD*, 2000.

[20] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug. 2004.

[21] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner. Imputation-boosted collaborative filtering using machine learning classifiers. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 949–950, New York, NY, USA, 2008. ACM.

[22] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner. Imputation-boosted collaborative filtering using machine learning classifiers. In *SAC*, pages 949–950, 2008.

[23] B. Ustun, W. Melssen, and L. Buydens. Facilitating the application of support vector regression by using a universal pearson vii function based kernel. 2006.

[24] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, editors, *SIGIR*. ACM, 2006.

[25] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 114–121. ACM, 2005.

[26] H. Yu and S. Kim. Svm tutorial — classification, regression and ranking. In G. Rozenberg, T. Bäck, and J. Kok, editors, *Handbook of Natural Computing*, pages 479–506. Springer Berlin Heidelberg, 2012.

[27] Z. Yu, X. Zhou, Y. Hao, and J. Gu. Tv program recommendation for multiple viewers based on user profile merging. *User Model. User-Adapt. Interact.*, 16(1):63–82, 2006.