A User-Study on Context-aware Group Recommendation for Concerts

Simen Fivelstad Smaaberg, Nafiseh Shabib, John Krogstie
Norwegian University of Science and Technology
Trondheim, Norway
smaaberg@stud.ntnu.no, {shabib, krogstie}@idi.ntnu.no

ABSTRACT

In this paper, we present a prototype of a group recommendation system for concerts. The prototype is context sensitive taking the user's location and time into account when giving recommendations. The prototype implements three algorithms to recommend concerts by taking advantage of what users have listened to before: a collaborative filtering algorithm (K-Nearest Neighbor), a Matrix Factorization algorithm, and a Hybrid approach combining these two. The usability of the prototype was evaluated using the System Usability Scale and a user centered evaluation was performed to evaluate the quality of recommendations. The results from the usability evaluation shows that users generally were satisfied with the usability of the prototype. The results from the Quality Evaluation shows that the K-Nearest Neighbor and Hybrid approach produces satisfactory results whereas the Matrix Factorization implementation was experienced to be a bit poorer. The users testing the prototype were generally satisfied with the quality of recommendations.

Keywords

collaborative filtering, group recommendation, context-aware

1. INTRODUCTION

Recommendation technology is becoming an increasingly important part of large systems such as Amazon.com and eBay.com, and also in the music industry for example Spotify, iTunes and Last.fm recommendations are used in to an increasing degree. A context-aware group recommendation system is a recommendation system that recommends items for groups of people instead of for a single person in the given context [14]. The group and context part adds additional challenges compared to a normal recommendation system for individual users [14]. A group of people is more dynamic than a single person. You have to consider how the group is formed, how unified recommendations for the whole group can be provided, and the dynamics within the group [6] [11]. Context in addition has many traits, but can be seen as external constraints that affects the recommendation process. This makes the algorithms more complicated. The purpose of this paper is to present a context-aware group recommendation system for concerts that takes the location and time of a user into account when making recommendations. This is done to show that traditional methods for Music Recommendation Systems can also be applied when concerts are recommended and extra context-variables have to be considered. Even though group recommendation systems have been explored, they are not as thoroughly investigated as recommendation systems for individuals. The same can be said for recommendation systems for concerts, and context-aware group recommendation systems, where limited existing research has been found, in particular on the perceived usability and quality of such solutions.
In the next section, we present the main approach, data model and algorithms. In Section 3, the experiments and evaluations is presented. Related work is described in Section 4, before we conclude in Section 5, pointing to future work.

2. DATA MODEL AND RECOMMENDATION ALGORITHM

Illustrating the problem of context-based group recommendation, we take the following scenario as an outset:

A group of friends is traveling to a big city to stay there for a week. Here they wants to attend a concert. Their tastes in music are quite different, so choosing what concerts to attend is a challenge. Moreover, they may not be familiar with all the bands playing and would like to have an application that give them recommendations concerning which concerts to attend based on the type of music they have listened to before and their personal musical preferences.

We consider the following requirements for designing context-aware group recommendation for concerts:

- Recommendations need to be based on the user's listening preferences
- The system should be location-aware (concerts close to a user are preferred)
- The system should be time-aware (not recommend concerts that already have taken place or concerts too far ahead in time)
- Context relaxation should be supported (recommendations for more widespread locations or time-period can be attempted if not enough concerts are found for the given location or time)

2.1 Data model

The main concepts used to support these requirements are depicted in the data-model in Figure. 1. Users have previously listened to music by existing artists. The artists

are in addition tagged (with musical categories), and one have information about the tags/interests of users. Based on listening and tagging history, user-similarity is calculated. Artist play concerts. The concerts take place on venues at a certain time and space (geographically located) in a particular city.

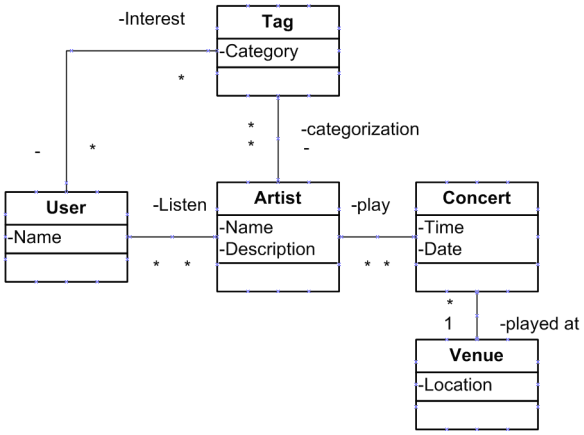

Figure 1: Conceptual Data Model

2.2 Context and Context relaxation

Context captures information that is not part of the database, such as the user location or the current time. A user would probably want to get recommendations for concerts in locations close to where he is located, and not get recommendations for concerts too far ahead in time (unless planning ahead for a later travel of course), or for concerts that already have taken place. Therefore the definition of these context parameters should be a central part of any concert recommendation system (`CRS`). In particular, a context parameter can be relaxed upwards by replacing its value by a more general one, downwards by replacing its value by a set of more specific ones or sideways by replacing its value by sibling values in the a context-hierarchy [13], which in our case would be an adjecent later day or a neighboring city. To enable for relaxation of the location parameter, the 100 concerts closest to the location specified is also fetched. This is done by utilizing the Haversine formula[1]. This formula can be used to estimate the shortest distance between two points on the earth surface[3]. In addition, concerts within 5 days of the date range specified is fetched to support relaxing of the date range parameter.

2.3 Listen Count Normalization

We have retrieved listening history from the Last.fm music discovery service. 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 on Last.fm dataset. Similarly to the approach taken by [4], for each user, U, its listening counts for each artist, A, is normalized using the Cumulative Distribution Function (CDF) of the artist listenings for U. The artists with a listening count falling within the first 10% of this distribution is assigned a rating of 10; the artists falling within the first 20% of the distribution is assigned a rating of 9; and so on until the artists within 90 to 100% of the distribution is assigned a rating of 1.

[1]http://en.wikipedia.org/wiki/Haversine_formula

2.4 Neighborhood Model

The K-Nearest Neighbor (KNN) algorithm was one of the first approaches used in user-based recommendation [5]. In our work, the KNN algorithm is split into two phases:

1. Filter dataset
2. Recommend concerts

2.4.1 Filter dataset

In a KNN approach, the K-Nearest Neighbors of u are used as a basis for recommendation. For simplicity reasons we state that a user that have not listened to any of the artists in A' cannot be considered by the algorithm. This can be done because a user that has not listened to any of the artists in A', could only contribute with a listening count of 0 to all of them, and therefore the user might be left out.

Since the set of artists that are considered in the algorithm has been reduced to the set of artists A' playing at one of the concerts in C', implicitly the set of users considered for the algorithm can be reduced to the set of users that have listened to one or more of the artists in A'.

$$U' = \{u' : \forall u' \in U \exists a \in A' listenedTo(u', a)\} \quad (1)$$

2.4.2 Recommend concerts

In a KNN algorithm, the K most similar users to u are found, and their ratings are used as a basis for recommendation. To find these similar users, we applied cosine similarity based on listening count of two users for each artist. So, the user vector $\mathbf{w}_i$ for a user $u_i \in U$ is defined as the vector of the users listening counts to each of the artists in A.

$$\mathbf{w}_i = \{listenCount(i, a) : a \in A\} \quad (2)$$

In a normal K-Nearest Neighbor algorithm the K users with the highest similarity would now be identified and used as a basis for recommendation. For the purpose of a `CRS`, this is not enough. Here, a rating for each of the concerts, c_i, in C' have to be predicted. Therefore, a 3 step process is undertaken for each of the concerts:

1. Find the K users, U'', with the highest similarity to u from the subset of U' that have listened to one or more of the artists performing at that concert.
2. Calculate the predicted rating for each of the artists a playing at the concert. $TotalSimilarity$ is defined as the sum of similarities to u from each of the users in U''. Each of the users u_i in U'' will contribute to the predicted rating with a percentage of $\frac{sim(u_i,u)}{totalSimilarity}$. The actual contribution is influenced by the rating given to a by u_i, so this is multiplied with $rating(u_i, a)$. The predicted rating for an artist i will then be:

$$artistRating_i = \sum_{j=1}^{n} \frac{sim(u_j, u) \times listenCount(u_j, a_i)}{totalSimilarity} \quad (3)$$

3. The overall predicted rating for the concert c_i as a whole for user u is given by the average of the predicted

ratings to each of the m artists performing at c_i.

$$KNNRating^u_{c_i} = \frac{\sum_{k=1}^{m} artistRating_k}{m} \quad (4)$$

2.5 Latent Factor Model

Similarly to [7], the $n \times m$ user-artist matrix M is reduced into a set of user vectors, V, where $V_i \in \mathbb{R}^f$ and artist vectors, B, where $B_i \in \mathbb{R}^f$. f is the number of latent factors to extract (dimensionality of the latent factor space). In this work, the user-artist matrix consists of the normalized listen counts for all of the users in U and the artists in A. To approximate a user u's rating for an artist a, $\hat{r}_{ua}$, the dot product between u's and a's latent factor vectors $V_u B_a$ is performed. As [7] says: this dot product "captures the interaction between user u and item i - the users' s overall interest in the item's characteristics".

$$\hat{r}_{ua} = B_a^T V_u \quad (5)$$

We will refer to this model as $PureSVD$. It uses $f = 64$ features which are optimized by running over 120 iterations. The implementation is based on Timely Developments[2] implementation of the algorithm.

The overall predicted rating for the concert c_i as a whole for user u is given by the average of the predicted ratings to each of the m artists performing at c_i.

$$mfRating^u_{c_i} = \frac{\sum_{a=1}^{m} \hat{r}_{ua}}{m} \quad (6)$$

2.6 Hybrid Model

The predictions given by the algorithms in the previous two sections are in this phase aggregated to produce the final top N concerts to return to the user. For each of the concerts, c_i, in C the final rating for the concert for u, $r_{uc_i} \in R_u$ is given by:

$$\hat{r}_{u_i c} = \frac{mfRating^u_{c_i} + knnRating^u_{c_i}}{2} \quad (7)$$

The N concerts with the highest rating r_{uc_i} in R are selected and returned to the user.

2.7 Aggregation strategy

In this work, an *average* aggregation strategy (which computes the group preference for an item as the average of group members' preferences for that item) is used to aggregate individual ratings into a group rating for a concert. Since, in a music recommendation system we have to utilize implicit feedback, there is no such thing as a negative preference. For example, a listen count of 0 does not necessarily mean that a user does not like the artist, just that the user has not listened to the artist before. The user might like the artist, but he has not discovered it, or he might dislike it. Therefore, it is impossible to know for certain how to interpret a listen count of 0. Similarly, a low listening count may not mean that a user does not like the artist, he might just have discovered the artist or just joined the system. Again, it is impossible to know. Thus, we can safely assume that Least Misery (which computes the group preference for an item as the minimum among all group members' preferences for that item) in an aggregation method would not be applicable thus we used the *average* strategy.

[2]http://www.timelydevelopment.com/demos/NetflixPrize

3. EXPERIMENTS

We evaluate our group recommendation system from two major angles. First, from the *usability* perspective (Section 3.1), and second *quality* perspective (Section 3.2).
We implemented our prototype system using Java and MySQL for the back end. The front end was developed in JavaScript and HTML5, and is based on the Durandal.js[3] Model View Viewmodel framework.

Dataset description: We use the Last.fm dataset for evaluation purposes. Last.fm has become a relevant online service in music based social networking. In our particular `CRS` the data was fetched using Last.fm's publicly available API[4].The dataset as seen in Table 1 consists of 2,891 concerts in Vancouver, New York, London, Oslo, and surrounding areas, between 18. February 2014 and 6. June 2014. The dataset was built by first fetching concerts within a 100km radius from the specified cities. Then, information about the artists performing at those concerts were fetched. Users that have listened to the artists found are then fetched, before the 30 most listened to artists for each user are fetched and saved. In addition to these data, information about the venue that each concert is held at and the most used tags for each artist is stored. When a new user was created where no existing data was present in Last.fm, he would need to rate at least 5 artist that are registered in Last.fm. In the quality experiments below, we have looked upon differences when providing 5 or 10 ratings.

Property	Count
Users	25720
Artists	80877
Concerts	2891
Listening counts	769370
Tags	159348
Tags for artists	1358715
Artist concert participation	6845
User similarities	17025096
Venues	596
User features	17025096
Artist features	5085312

Table 1: Dataset properties

3.1 Usability Experiment

In this work, to evaluate the usability, we recruited 15 participants to use the system and answer three questionnaires, the System Usability Scale (SUS), an Application Specific survey (AS) and a questionnaire to gather Background Information (BI). The result view of the system can be seen in Figure 2 giving an indication on the look and feel of the system.

The System Usability Scale is a "reliable, low-cost usability scale that can be used for global assessments of systems usability" [2]. It gives a global view of subjective assessments that indicates how users agree or disagree with the

[3]http://durandaljs.com/
[4]http://www.last.fm/api

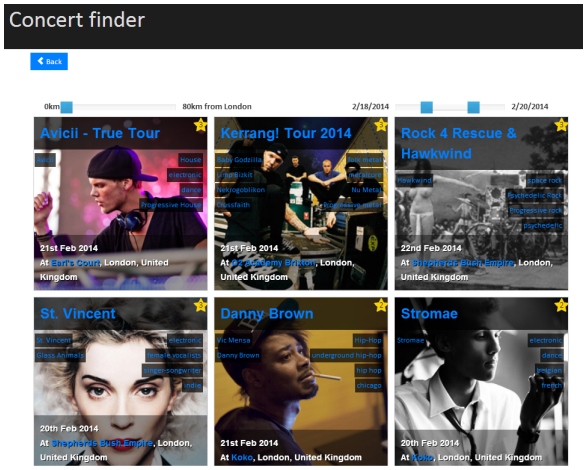

Figure 2: Result view of the prototype

statement. Nielsen suggests that 5 users are enough to find the majority of usability problems of a system, those 5 participants could reveal about 80% of all usability problems [9]. In general, one should run usability tests with as many participants that schedules, budgets, and availability allow. On this basis we are confident that with our 15 users, we have covered the main usability issues of the application.

3.1.1 *Results*

The results from the SUS survey yielded a SUS score of 79.83. [1] proposes an adjective rating scale to help determine what SUS scores actually mean. According to these adjective ratings, a SUS score of 79.83 would fall into somewhere between *Good* and *Excellent*. There is no absolute score when it comes to usability evaluations, but a score of 79.83 is a good indication on that the users found the usability of the prototype satisfactory. The results from the Application Specific survey (AS) showed that 66% of the participants believed that the would use this application in future. 87% of the participants answered either *OK* or *Satisfied* when asked how satisfied they were with the quality of recommendation from these music recommendation services, although the real quality evaluation in a group setting was postpone to the quality evaluation reported below. Concrete improvement proposals gathered were used to develop the second version of the system where more detailed quality experiment was undertaken

3.2 Quality Experiment

To evaluate the quality of recommendations from the improved system, two groups consisting of two and three people respectively were asked to find recommendations both individually and in a group setting, for different dates and places, and to rate how satisfied they were with the given recommendations. For this purpose, we showed three different lists where each list was the result of using the three different algorithms (k-NN algorithm, MF algorithm, the hybrid approach). Each of the lists are given "random" case ids and placed in a random order. The participants were asked to find recommendations individually, in a group of two people, and in a group of three people, for two different timespans (18/02/2014 – 03/03/2014 and 05/03/2014 – 09/07/2014), and two different cities (London and New York). When the second group were asked to find recommendations for a group of 3 people, a user from the first group were added to the recommendation process. For each step, they rated each of the algorithms on how satisfied they were with the recommendations given on a scale from 1-5, where 1 is *Very dissatisfied* and 5 is *Very satisfied*.

Algorithm	Number of selections	Percentage
Matrix Factorization	7	17.5%
k-Nearest Neighbor	16	40.0%
Hybrid approach	17	42.5%

Table 2: Preferred algorithm selection by users

3.2.1 *Results*

As seen in Table 2, the MF algorithm were overall picked as giving the most appealing results 7 out of 40 times, the kNN algorithm 16 out of 40 times, and the hybrid approach in 17 out of 40 cases. Overall, the kNN algorithm received an average rating of 3.72 in the 40 responses, the Hybrid approach 3.62, and the MF algorithm an average of 2.87 as seen in Table 3. Table 4 shows statistics when recommendations were given for groups consisting of 1, 2 and 3 users respectively. From these results there is a clear trend that the kNN and the hybrid approach tend to produce more satisfying recommendations than the MF approach as the average ratings given to the two are generally lower, and they were picked as the favorite algorithms significantly more. An overall average rating of 3.72 and 3.62 out of 5 from the kNN and Hybrid approaches respectively, indicates that the participants were reasonably satisfied with the results given. In general, recommendations given for users created based on 10 of the user's favorite artists, produced more satisfying results than when 5 artists were used in the user creation process. Moreover, by increasing the number of users in a group from two to three users user satisfaction is decreasing.

Algorithm	Average rating	Variance	Standard Deviation
Matrix Factorization	3.13	0.73	0.85
k-Nearest Neighbor	2.28	0.92	0.96
Hybrid approach	2.38	0.75	0.87

Table 3: Overall Average statistics per algorithm

3.3 Insight about Serendipity in Concert Recommendation Systems

Serendipity is concerned with the novelty of recommendations and in how far recommendations may positively surprise the user [12] and it has received increased attention that recommendation system should provide novel and serendipitous recommendations. The emphasis should be put on the lesser known artists, the *long tail* of the listen count curve. However during the development and testing of this prototype it was observed that a full focus on this may not be the best approach for a `CRS`. Our findings show that people tend to prefer to go to concerts with artists they

	Average rating			Variance			Standard Deviation		
Algorithm	1	2	3	1	2	3	1	2	3
Matrix Factorization	2.7	3.3	3.1	0.46	1.12	0.77	0.67	1.06	0.88
k-Nearest Neighbor	4.2	3.7	3.5	0.62	1.12	1.17	0.78	1.06	1.08
Hybrid approach	3.7	3.6	3.4	0.68	0.93	0.93	0.82	0.97	0.97

Table 4: Statistics when recommendations were given for groups consisting of 1, 2 and 3 users respectively

kNN		MF	
Artist	# of listeners	Artist	# of listeners
Avicii	548	Arctic Monkeys	2388
Katy Perry	676	Lorde	554
Arctic Monkeys	2388	Beyoncé	585
Disclosure	535	Metronomy	418
Kanye West	1578	Cut Copy	378
Nine Inch Nails	1270	Alkaline Trio	383
The National	1687	Panic! at the Disco	
Drake	712	Slowdive	308
Interpol	784	Katy Perry	676
Arcade Fire	2165	Pretty Lights	234
Average	**1234**	**Average**	**632**

Table 5: Number of listeners for the top artist playing at the top 10 concerts between 18/02/2014 and 17/07/2014 in London for user *simensma*

already familiar with and the concert scene might not the place were people try to be adventurous and discover new music, it is easier, more convenient, and cheaper to discover and becoming familiar with new artists first, before deciding to attend a concert with them. This might be one of the causes why the kNN and Hybrid approaches received better ratings from the test users when it came to quality of recommendations, as collaborative filtering (CF) approaches tend to have a popularity bias causing the more popular artists to be recommended. An example of this can be seen in Table 5 where the top artist and how many users have listened to them for the 10 top concerts recommended for the user *simensma* in London between 18/02/2014 and 17/07/2014 can be seen. The 5 most frequently used tags to describe *simensma*'s top artists are *electronic*,*house*,*dance*,*indie*, and *electro house*. On average, 1234 users had listened to each of the artists recommended by the kNN algorithm whereas 632 users on average had listened to each of the artists recommended by the MF algorithm.

3.4 Threat to validity

The quality evaluation was performed with only two groups of 2 and 3 people. This low number of participants means that each participant had a very large impact on the results. The statistics produced when a user was created with 5 and 10 favorite artists, were based on n = 10 samplings each; the same was the case with the statistics produced for the results with varying group sizes. By looking at the top tags used for the artists each of the users registered, it is apparent that the users' taste in music are quite different as they share few top tags amongst them. However, because of the low number of users and sample sizes, even with this diversity, it cannot be said that these five users are representative for the whole potential user base, and therefore, further testing should be performed to measure the Quality of Recommendations created by the prototype. Even though more testing is needed, there still is a strong indication that the KNN and Hybrid approaches perform better than the MF approach as suggested with a sample size of n = 40. Similarly, it can be said that the five users testing the prototype were reasonably happy with the resultss.

4. RELATED WORK

Group Recommendation Systems try to provide recommendations to a group of people instead of a single individual. There are two main approaches of accomplishing this: calculating recommendations individually for each of the members of the group, and then aggregating the individual results, or merging the preferences of each of the members of the group, and then providing one set of recommendations based on the merged profile [10][8]. In either of the approaches, there are many ways this merging can be accomplished [8]. This includes least misery, average, and average without misery. The choice of aggregation strategies should be decided based on the problem you are trying to solve, as there is no universal best strategy that works in all cases. As argued above, we choose an average aggregation strategy. Recommendation Systems for Music (MRS) have increasingly become an important part of music services. Services such as iTunes, Spotify, last.fm and Pandora all incorporate music recommendations centrally in their user interface. With an ever growing collection of music, these services compete in finding new and innovative ways on how users can discover new music. Celma [4] identifies three use cases typical for a MRS: neighbor finding, playlist generation and artist recommendation. Neighbor finding consists of finding users with a similar taste in music as you. Playlist generation usually means finding songs to recommend for a user, but instead of just returning the top N songs, songs that go well together are preferred. Artist recommendation usually consists of finding artists based on a user's profile, be it the artist with the highest predicted rating or novel artists. Different services apply a variety of techniques when it comes to the recommendation process [15]. Some of them are acoustic analysis, text analysis, editorial review, and the use of activity data. This diversity indicates that people have different ways to think about music. Enthusiasts and savants might prefer to try out new and little known artists, whereas casual users might prefer well known artists and the latest 'big hits'. With such a diverse set of expectations, creating a music recommendation system that works well for all of them is challenging. In general these techniques are provided for creating recommendations for individual users, little work being done to support groups of users.

5. CONCLUSION

In this paper, a prototype of a context-aware group recommendation system for concerts was presented. The prototype implemented three different algorithms, a Matrix Factorization algorithm, a k-Nearest Neighbor algorithm and a Hybrid approach of the two. The usability of the prototype was evaluated using the System Usability Scale (SUS) and an Application Specific Survey (AS). 15 people were asked to undertake these surveys. In total, the prototype got a SUS score of 79.83 which is a good indication on that the users found the usability of the prototype satisfactory. However, the comments from the free text answers shows that there where room for improvements. The AS mainly focused on the usability of the context relaxation part of the prototype, to find out if it was easy to find concerts close to the parameters specified when it comes to time and location. The results from the AS showed that the users in general were satisfied with how this process worked. The goal for this prototype was to recommend concerts to a user within the location and timespan given that the user could be interested in attending. To evaluate how well this was achieved, a Recommendation Quality Evaluation(QE) was undertaken with two groups consisting of 2 and 3 people respectively. Through a range of scenarios, the groups were told to find recommendations for the dates and location asked about, and for each algorithm, rate how satisfied they were with the results. The results from the QE showed that the users generally were satisfied with the KNN implementation and the Hybrid approach, whereas they were less satisfied with the MF approach. The QE was also undertaken to see how different group sizes affected the quality of recommendations. The results showed that the users became less satisfied when the number of members in the group increased from one to two and three respectively, which is to be expected as different preferences has to be taken into account in larger groups. However, the QE was only performed with five participants, so there is a need for an evaluation with more participants to able to draw any further conclusions. Another way to go from here is to have a look at the context-aware part of the application. Is there any benefit in making relaxation of context an implicit part of the algorithm instead of something performed by the user explicitly? How would other context variables, such as listen recency affect the satisfactions when recommending concerts?

6. REFERENCES

[1] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[2] J. Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189:194, 1996.

[3] C. A. Cassa, K. Iancu, K. L. Olson, and K. D. Mandl. A software tool for creating simulated outbreaks to benchmark surveillance systems. *BMC Medical Informatics and Decision Making*, 5(1):22, 2005.

[4] O. Celma. *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space*. Springer, 2010.

[5] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[6] D. R. Forsyth. *Group dynamics*. Brooks/Cole, Pacific Grove, Calif., 2. edition, 1990. Donelson R. Forsyth. graph. Darst ; 24 cm. Fräüher u.d.T.: An introduction to group dynamics.

[7] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[8] J. Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Model. User-Adapt. Interact.*, 14(1):37–85, 2004.

[9] J. Nielsen. *Usability engineering*. Elsevier, 1994.

[10] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Space efficiency in group recommendation. *VLDB J.*, 19(6):877–900, 2010.

[11] N. Shabib, J. A. Gulla, and J. Krogstie. On the intrinsic challenges of group recommendation. In *RSWeb@RecSys*, 2013.

[12] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[13] K. Stefanidis, E. Pitoura, and P. Vassiliadis. On relaxing contextual preference queries. In *MDM*, pages 289–293, 2007.

[14] K. Stefanidis, N. Shabib, K. Nørvåg, and J. Krogstie. Contextual recommendations for groups. In *ER Workshops*, pages 89–97, 2012.

[15] B. Whitman. How music recommendation works and doesn't work. `http://notes.variogr.am/post/37675885491/how-music-recommendation-works-and-doesnt-work`, 2013. Accessed: 2013-04-27.