Chapter 3
Evaluating Group Recommender Systems*

C. Trattner, A. Said, L. Boratto and A. Felfernig

Abstract In the previous chapters, we have learned how to design group recommender systems but did not explicitly discuss how to evaluate them. The evaluation techniques for group recommender systems are often the same or similar to those that are used for single user recommenders. We show how to apply these techniques on the basis of examples and introduce evaluation approaches that are specifically useful in group recommendation scenarios.

3.1 Introduction

Evaluating group recommenders is intrinsically related to evaluation techniques used for single user recommenders [11, 15]. There are two types of *evaluation protocols*: (1) *offline* and (2) *online evaluation*.

Offline evaluation is based on the idea of estimating the prediction quality of an algorithm using *datasets* that include user $\times$ item evaluations (ratings). These datasets are typically divided into *training* and *test sets* with a split of, for example, 80% training data and 20% test data. Such settings are used

Christoph Trattner
University of Bergen, Norway. e-mail: christoph.trattner@uib.no

Alan Said
University of Skövde, Sweden. e-mail: alansaid@acm.org

Ludovico Boratto
EURECAT, Spain. e-mail: ludovico.boratto@acm.org

Alexander Felfernig
Graz University of Technology, Austria. e-mail: alexander.felfernig@ist.tugraz.at

* This is a pre-print of the chapter "Evaluating Group Recommender Systems" published in the book "Group Recommender Systems: An Introduction" authored and edited by Felfernig, A., Boratto, L., Stettinger, M. and Tkalcic, M. The book is available at http://www.springer.com/us/book/9783319750668.

for the evaluation of a recommendation algorithm in the light of given *evaluation metrics* on the basis of repeated sampling and cross validation [4, 8]. Since datasets are typically derived from recommender systems for individual users [14], *datasets for groups* have to be *synthesized* to be applicable to the evaluation of group recommenders [2].

Online evaluation is based on the idea of using user study techniques to evaluate an algorithm, a user interface, or a whole system online [16]. Over the past few years, this approach has lagged behind offline evaluation, due to higher efforts and the lack of standardized evaluation frameworks [16, 19]. *Lab studies* (as one type of online evaluation) involve the recruitment of study participants who are then engaged in tasks based on two kinds of designs: (1) *within-subjects* study design (each subject is assigned to a *set of conditions*) or (2) *between-subjects* study design (each subject is assigned to *exactly one condition*). Online evaluations in the form of lab studies in the context of group recommender systems have been conducted, for example, by Zapata et al. [30], De Pessemier et al. [7], Masthoff et al. [17], and Stettinger et al. [25]. As an alternative to lab studies, which are often quite costly, recent research in recommender systems started to use *crowd sourcing platforms* as a source of user feedback [6, 28]. A major challenge in this context is to assure the understanding of the defined tasks and to include quality assurance mechanisms to avoid low-quality feedback [1]. Finally, *naturalistic studies* often employ some kind of *A/B testing* where the system is used *as is* without any interventions or predefined tasks. A/B in this context refers to different user populations unknowingly assigned to different system versions in which some condition has been changed. Naturalistic studies in the context of group recommender systems have been conducted, for example, in Sanchez et al. [20].

Independent of the type of online evaluation protocol used, quantitative post-hoc analysis is typically employed to identify differences between interfaces, algorithms, and systems. Apart from the standard evaluation metrics (as discussed in the following), metrics such as *number of clicks*, *time needed to complete a task*, and *dwell time* are also employed to measure system efficiency. Based on the exact evaluation protocol defined, a set of recommendation metrics can then be used to estimate the performance of the recommender system.

In the following, we *first* focus on *accuracy metrics* which compare recommendations determined by a recommender system with a predefined set of real-world user opinions (also known as *ground truth*).[2] Depending on the underlying goal, accuracy can be measured on the basis of: (1) *classification metrics* that evaluate to which extent a recommender is able to determine items of relevance (interest) for the user, (2) *error metrics* that evaluate how well a recommender predicts ratings, and (3) *ranking metrics* that help to evaluate how well a recommender predicts the importance ranking of items.

[2] For an in-depth discussion of evaluation metrics for single user recommenders we refer to Gunawardana and Shani [24].

Second, we discuss a couple of *group recommendation-related metrics* that go beyond accuracy measurement.

3.2 Classification Metrics

Arguably, the most common classification metrics used in recommender systems are *precision* and *recall*. These metrics are often applied in *offline* evaluation scenarios where recommendation algorithms are trained using a portion of the available data for learning purposes and are then evaluated by comparing predictions to a withheld part of the data ('holdouts' constituting the test set). In the following, we will briefly explain the usage of precision and recall metrics in group recommendation scenarios. Table 3.1 contains (1) *user rating data* (evaluations of items already consumed by the members of groups g_1, g_2, and g_3) where each group consists of three users, and (2) *predictions of item ratings* (for items t_1 and t_2). For simplicity, we assume that each group member provided a rating for each item consumed by her/him.

groups	group members	ratings $r(u_i, t_j)$			predictions $\hat{r}(u_i, t_j)$		
		t_1	t_2	...	t_1	t_2	...
g_1	u_1	4.5	2.5	...	3.4	3.8	...
	u_2	3.5	4.5	...	3.7	4.4	...
	u_3	4.5	4.0	...	4.4	3.9	...
g_2	u_4	3.5	2.5	...	3.8	2.6	...
	u_5	4.0	4.5	...	3.7	4.4	...
	u_6	4.5	3.5	...	4.5	3.7	...
g_3	u_7	4.5	3.5	...	3.4	3.8	...
	u_8	3.5	2.5	...	3.7	4.4	...
	u_9	4.0	3.5	...	4.4	3.9	...
...	...	...	...	...	...	...	...

Table 3.1: *Ratings* $r(u_i, t_j)$ and *predictions* $\hat{r}(u_i, t_j)$ for items t_1 and t_2.

The user-individual ratings and predictions are aggregated into (1) a *group rating* $r(g_i, t_j)$ and (2) corresponding *group predictions* $\hat{r}(g_i, t_j)$ determined by an *aggregated predictions* based group recommender system (see Table 3.2).[3] In a typical group recommendation scenario, a random set of *group-level item ratings* is withheld and used as test set. In our example, we assume for simplicity that for groups g_1, g_2, and g_3, the ratings for item t_1 and t_2 have been selected as 'holdouts'. The rating predictions $\hat{r}(g_i, t_j)$ (assumed to be provided by a group recommender) are depicted in the two rightmost columns of Table 3.2.

[3] Predictions are determined using rating data not used as test cases.

groups	ratings $r(g_i, t_j)$		predictions $\hat{r}(g_i, t_j)$	
	t_1	t_2	t_1	t_2
g_1	4.2	3.7	3.8	4.0
g_2	4.0	3.5	4.0	3.6
g_3	4.0	3.2	3.8	4.0

Table 3.2: Example *test set*: *group ratings* $r(g_i, t_j)$ and *group predictions* $\hat{r}(g_i, t_j)$ where $r(g,t) = \frac{\Sigma_{u \in g} r(u,t)}{|g|}$ and $\hat{r}(g,t) = \frac{\Sigma_{u \in g} \hat{r}(u,t)}{|g|}$. For groups g_1, g_2, and g_3, the item ratings of t_1 and t_2 are considered as elements (holdouts) of the test set.

On the basis of the entries in Tables 3.1 and 3.2, we will now sketch the application of the classification metrics *precision* and *recall*.

Precision is the fraction of the number of *relevant recommended items* (true positives) in relation to the *total number of recommended items*. *Recall* is the fraction of the number of *relevant recommended items* in relation to the *number of all relevant items*. Both metrics are commonly expressed at a certain level k where k is the length of the list of recommended items. For example, $precision@1 = 1$ indicates that one item was recommended and this item was deemed to be a relevant recommendation. Furthermore, $precision@2 = 0.5$ denotes a situation where in a list of two recommended items only one is deemed to be a relevant recommendation (true positive). The precision of a group recommender that recommends k items to a group g can be defined as follows where $predicted_k(g)$ denotes a list of k items recommended to group g and $relevant(g)$ represents all items relevant for g. Definitions of *precision* and *recall* are given in Formulae 3.1 and 3.2.

$$precision@k(g) = \frac{|predicted_k(g) \cap relevant(g)|}{k} \tag{3.1}$$

$$recall@k(g) = \frac{|predicted_k(g) \cap relevant(g)|}{|relevant(g)|} \tag{3.2}$$

The calculation of precision and recall is sketched in Table 3.3 – it is based on the test dataset defined in Table 3.2. For the purpose of our example, we define an item to be relevant, if the corresponding group rating > 3.5 (the *relevance threshold*). For example, $precision@2(g_1) = 1$ since both items predicted to be relevant (rating > 3.5) are deemed as relevant by group g_1. The overall precision and recall values for the *test set* (see Table 3.2) are determined by integrating the group-specific prediction and relevance counts, for example, 6 predicted items of 4 correctly predicted items result in an overall precision of 0.67 (see Table 3.3).

Precision and *recall* can be used in *aggregated predictions* as well as in *aggregated models* based group recommenders (see Chapter 2) since both evaluation metrics are applied to the recommendation result, i.e., are independent

groups	predicted	relevant	precision@2	recall@2
g_1	2	2	1.0	1.0
g_2	2	1	0.5	1.0
g_3	2	1	0.5	1.0
overall	6	4	0.67	1.0

Table 3.3: *Precision* and *recall* values in example scenario.

from the underlying aggregation approach. The same holds for *content-based*, *constraint-based*, and *critiquing-based recommendation*. In these scenarios, the group recommender determines an overall item evaluation (similarity between user and item in content-based filtering, user-specific item utility in constraint-based recommendation, and similarity between candidate and reference item in critiquing-based recommendation) which is then used for estimating item relevance. Consequently, a threshold similar to the one used in our example can be applied.

As will be discussed in Chapter 7, some group recommenders operate on item *packages* and *parameters* [9, 10]. Package recommendations can be evaluated similarly to single item recommendations – a difference in this regard is that package items get recommended at the same point of time whereas single items are recommended at different points of time. Recommendations of configurations [10] (see Chapter 7) consist of parameter settings related to requirements of a single user (or a group). In this context, *precision* can be defined as the share of *correctly predicted parameter values* compared to the *total number of predicted parameter values*. Furthermore, *recall* can be defined as the share of *correctly predicted parameter values* compared to the *total number of relevant parameter values* (see Formulae 3.3 and 3.4).

$$precision(g) = \frac{|predictedvals(g) \cap relevantvals(g)|}{|predictedvals(g)|} \tag{3.3}$$

$$recall(g) = \frac{|predictedvals(g) \cap relevantvals(g)|}{|relevantvals(g)|} \tag{3.4}$$

An example of the calculation of *precision* and *recall* for group g in a group-based configuration scenario is sketched in Table 3.4. In this example, $precision = \frac{2}{4} = 0.5$ (2 correct predictions out of 4) and $recall = \frac{2}{5} = 0.4$. Our assumption is that parameter values for parameters par_1 .. par_4 have been predicted by a (group) recommender system.

Remark. Note that this approach to determine *precision* and *recall* can also be applied to evaluate the predictive quality of diagnosis algorithms [12] (see Chapters 1 and 2). In this case, *precision* can be regarded as the share of correctly predicted diagnoses compared to the total number of predictions. Likewise, *recall* is the share of correctly predicted diagnoses compared to the total number of relevant ones.

group g	relevant	predicted
$par_1 = a$	$\surd$	$\surd$
$par_1 = b$	$\times$	$\times$
$par_1 = c$	$\times$	$\times$
$par_2 = 1$	$\surd$	$\times$
$par_2 = 2$	$\times$	$\surd$
$par_3 = a$	$\surd$	$\times$
$par_3 = b$	$\times$	$\surd$
$par_4 = u$	$\surd$	$\surd$
$par_4 = v$	$\surd$	$\times$
$par_4 = w$	$\times$	$\times$

Table 3.4: Relevant (indicated by $\surd$) and predicted (indicated by $\surd$) parameter values in a group-based configuration scenario. par_4 is assumed to be multi-valued, i.e., can have more than one value at a time.

3.3 Error Metrics

Error metrics can be used to measure the error made by a recommender system to predict a rating of an item. The underlying assumption is that the smaller the error, the better the evaluated algorithm. A basic means of measuring prediction errors is *mean absolute error* (MAE) (see Formula 3.5). A detailed discussion of error metrics is given, for example, in Shani and Gunawardana [24]. In Formula 3.5, R_g denotes the set of ratings of group g contained in the test set (see Table 3.2).

$$MAE(g) = \frac{\Sigma_{r(g,t) \in R_g} |r(g,t) - \hat{r}(g,t)|}{|R_g|} \tag{3.5}$$

The determination of the MAE value for the rating predictions in the test set shown in Table 3.2 is depicted in Table 3.5. The overall MAE value for a test set can be determined by averaging group-specific MAE values.

groups	MAE
g_1	$\frac{0.4+0.3}{2} = 0.35$
g_2	$\frac{0.0+0.1}{2} = 0.05$
g_3	$\frac{0.2+0.8}{2} = 0.5$
$overall(AVG)$	0.3

Table 3.5: Mean absolute error (MAE) values determined on the basis of the rating information included in Table 3.2.

Similar to *precision* and *recall*, MAE can be used in the context of *aggregated predictions* and *aggregated models* based group recommenders. Given a function that estimates user $\times$ item ratings, this metric can also be applied in *content-based*, *constraint-based*, and *critiquing-based recommendation*. Furthermore, MAE can be applied to scenarios such as *package recommendation* and *group-based configuration* (see Chapter 7). An example of determining MAE in *configuration scenarios* is given in Table 3.6. For simplicity, we assume that parameter values are numeric.[4]

group g_1	relevant	predicted	MAE
$par_1(1,2,3)$	1	3	$\|1-3\| = 2$
$par_2(1,2)$	2	2	$\|2-2\| = 0$
$par_3(1,2,3,4)$	2	3	$\|2-3\| = 1$
$overall(AVG)$	-	-	1.0

Table 3.6: Determining MAE in configuration scenarios (measuring the distance between predicted parameter values and those regarded as relevant ones).

Remark. Note that the usage of these metrics in the context of recommendation scenarios has declined as other types of metrics such as classification-based methods started to dominate. Recommendation is often interpreted as ranking problem.

3.4 Ranking Metrics

Ranking-dependent metrics do not only take into account item relevance but also the item position in a recommendation list. An example of such a metric is *discounted cumulative gain* (DCG) which is based on the idea that items appearing lower in a recommendation result should be penalized by downgrading relevance values logarithmically (see Formula 3.6). In Formula 3.6, k denotes the number of recommended items and $relevance(t_i, g)$ returns 1 if item t_i (at position i) is relevant for group g, and 0 otherwise.

$$DCG@k(g) = \Sigma_{i=1..k} \frac{2^{relevance(t_i,g)} - 1}{log_2(1+i)} \tag{3.6}$$

An example of the application of discounted cumulative gain (DCG) is provided in Table 3.7. The relevance values are derived from the group rating values of Table 3.2 ($relevance = 1$ if $r(g,t) > 3.5$, 0 otherwise). The more relevant items are included at the beginning of a list of k recommended items, the higher the DCG value. Since DCG operates on lists of ranked items, it can be applied to *collaborative filtering* as well as *content-based*, *constraint-based*, and *critiquing-based recommendation*. The overall DCG value for a test set is based on averaging group-specific DCG values.

[4] For a discussion of the handling of symbolic parameter values we refer to [26].

groups	*relevance*		$DCG@k$
	$pos_1 : t_1$	$pos_2 : t_2$	
g_1	1	1	$\frac{1}{1} + \frac{1}{1.6} = 1.625$
g_2	1	0	$\frac{1}{1} + \frac{0}{1.6} = 1$
g_3	1	0	$\frac{1}{1} + \frac{0}{1.6} = 1$
overall $DCG(AVG)$			1.21

Table 3.7: Example application of *discounted cumulative gain* (DCG).

If the length of recommendation lists for groups vary, i.e., there is no fixed k that reflects the number of recommended items, DCG has to be normalized by setting DCG in relation to the *ideal discounted cumulative gain* ($iDCG$) – see Formula 3.7. The resulting value is used to determine the *normalized discounted cumulative gain* ($nDCG$) – see Formula 3.8.

$$iDCG@k = \Sigma_{i=1..k} \frac{1}{log_2(1+i)} \tag{3.7}$$

$$nDCG@k(g) = \frac{DCG@k(g)}{iDCG@k} \tag{3.8}$$

In the line with the previously discussed evaluation metrics, DCG can be used in the context of *aggregated predictions* as well as in the context of *aggregated models*-based group recommendation. DCG can also be applied to *package recommendation* (see Chapter 7) by evaluating the predictive quality with regard to different item types, and by aggregating type-individual DCG values into an overall DCG value.

When evaluating *sequences of recommended items*, not only the position of the item (the later the worse) but also the position of the item compared to the position selected by the group, is of relevance. In this context, a simple approach is to compare items at individual positions in the sequence of recommended items with the sequence chosen by a group (k denotes the length of the sequence). Such an evaluation can be performed, for example, on the basis of Kendall's τ (see Formula 3.9).

$$\tau(g) = \frac{|concordantpairs| - |discordantpairs|}{k \times \frac{(k-1)}{2}} \tag{3.9}$$

An example of the application of Formula 3.9 is depicted in Table 3.8. A recommended item sequence (RS) gets compared with an item sequence finally chosen by a group (CS) – the ground truth. In this example, 4 concordances are against 6 discordances. A concordant pair is (t_1, t_3) since these items are mentioned in the same order in both sequences (RS and CS). In contrast, (t_2, t_4) is an example of a discordant pair since the order of mentioning differs (t_4 before t_2 in RS and t_2 before t_4 in CS). Consequently, $\tau = -0.2$ (on a scale of -1 .. +1).

position	1	2	3	4	5
RS	t_4	t_2	t_5	t_1	t_3
CS	t_1	t_2	t_5	t_3	t_4

Table 3.8: Example application of *Kendall's* τ to sequence evaluation (RS = recommended sequence, CS = chosen sequence).

3.5 Coverage and Serendipity

In the context of recommender systems, *coverage* can be considered from different points of view – see, for example, Ge et al. [13]. *User coverage* can be interpreted as the number of users for whom at least one recommendation can be determined. For group recommendation, we introduce the term *group coverage* (GC) which represents the share of groups for whom at least one group recommendation could be identified (see Formula 3.10). No recommendations for a group can be determined in situations where, for example, the aggregated item ratings are below a certain threshold (collaborative filtering), or where all the given group requirements do not allow a recommendation (which could be the case in constraint-based recommendation scenarios).

$$GC = \frac{|groupswithprediction|}{|groups|} \tag{3.10}$$

Catalog coverage (CC) serves the purpose of analyzing which items from a catalog get recommended to users (groups). It represents the share of items that were recommended to users (groups) at least once, compared to the total number of items contained in the item catalog (see Formula 3.11).

$$CC = \frac{|recommendeditems|}{|catalogitems|} \tag{3.11}$$

Serendipity, in the context of recommender systems, is defined as something surprising or unexpected that a user might not have seen before. A corresponding measure of *serendipity* (SER) is proposed in Ge et al. [13] (see also Formula 3.12). In this context, $RS(g)$ denotes the (useful) recommendations for group g determined by a group recommender system and $PM(g)$ denotes the recommendations generated by a primitive prediction model (e.g., based on item popularity). The overall SER value can be derived by averaging group-specific SER values.

$$SER(g) = \frac{|RS(g) - PM(g)|}{|RS(g)|} \tag{3.12}$$

3.6 Consensus and Fairness

Consensus can be regarded as a measure that evaluates to which extent group members have established an agreement with regard to their item preferences – see, for example, [5, 23]. In collaborative filtering, consensus can be measured in terms of the pairwise distances between the item-t ratings $r(u_i, t)$ of the individual group members u_i (see Formula 3.13) where r_{max} (the maximum rating possible) is used as normalization factor.

$$consensus(g,t) = 1 - \frac{\Sigma_{(u_i,u_j)\in g(i\neq j)}|r(u_i,t) - r(u_j,t)|}{|g| \times (|g|-1)/2 \times r_{max}} \tag{3.13}$$

If rating information is available, the same measure can be applied in *content-based*, *constraint-based*, and *critiquing-based recommendation*. In conversational recommendation, i.e., in constraint-based or critiquing-based recommendation, group members are engaged in an interactive process where they define and refine their preferences. In constraint-based recommendation, group members define their preferences as requirements whereas in critiquing-based recommendation critiques are used to represent preferences. In both cases, preferences can become inconsistent and have to be adapted so that a recommendation can be identified. In the context of conversational recommendation, we define consensus as the share of pairwise agreements (e.g., equal parameter value selections) between group members in relation to the total number of pairwise agreements and disagreements (conflicts) (see Formula 3.14).

$$consensus(g) = \frac{\#agreements(g)}{\#agreements(g) + \#conflicts(g)} \tag{3.14}$$

A simple example of evaluating the degree of consensus in conversational recommendation scenarios is shown in Table 3.9. In this example, we count the pairwise agreements and disagreements between $\{u_1, u_2, u_3\} = g$. The total number of disagreements is 4 and the total number of agreements is 5. Following Formula 3.14, the consensus level in this example is $\frac{5}{5+4} = 0.56$ (on a scale 0..1).

parameters	users			agreements	disagreements	consensus
	u_1	u_2	u_3			
$par_1(1,2)$	1	1	1	3	0	-
$par_2(1,2,3)$	1	2	1	1	2	-
$par_3(a,b)$	a	a	b	1	2	-
overall	-	-	-	5	4	0.56

Table 3.9: Example: determining *consensus* in conversational recommendation.

Since group recommender systems involve multiple stakeholders, they can give rise to *fairness* issues [3]. Burke [3] introduces the concept of *multi-sided fairness* where different stakeholder groups have different interests that should somehow be balanced. In such a context, fairness can be considered as *the extent of imbalance between group member specific utilities* [29]. If single items are recommended to a group by collaborative filtering, fairness can be specified, for example, on the basis of the share of item ratings above a relevance threshold th (see Formula 3.15).

$$fairness(g,t) = \frac{|\bigcup_{u \in g} : r(u,t) > th|}{|g|} \tag{3.15}$$

If we evaluate the t_1 ratings of group g_1 in Table 3.1 on the basis of Formula 3.15 assuming $th = 3.5$, the overall degree of fairness with regard to item t_1 is $\frac{2}{3} = 0.66$ (on a scale of 0..1). The fairness interpretation of Formula 3.15 primarily considers situations where single items are recommended to groups, i.e., this metric does not take into account situations where packages are recommended to groups [23]. Alternative definitions of *fairness* are the following.[5]

First, *m-proportionality* (see Formula 3.16) interprets fairness as the share of group members u_i with at least m items in the recommended (or selected) package for which u_i has a high preference [23]. In this context, g_p denotes the set of users for whom the m-proportionality condition holds.

$$fairness_{m-prop}(g) = \frac{|g_p|}{|g|} \tag{3.16}$$

An example ($m = 2$) of the calculation of a fairness estimate following the m-proportionality criteria is given in Table 3.10. The g_p value in this example is 2 since two users (u_1, u_3) each evaluated two items above the threshold rating of 3.5.

g_1	item ratings			rating threshold > 3.5	m-prop
	t_1	t_2	t_3		
u_1	5.0	4.0	1.0	2	1
u_2	4.0	3.0	2.0	1	0
u_3	4.5	5.0	5.0	3	1
u_4	3.5	3.0	5.0	1	0
$fairness_{m-prop}$	-	-	-	-	$\frac{2}{4} = 0.5$

Table 3.10: Evaluating fairness based on m-proportionality (m-prop) where $m = 2$.

[5] See also Chapter 6.

Second, *m-envy-freeness* (see Formula 3.17) interprets fairness as the share of group members u_i with at least m items for which u_i is in the top $x\%$ item ratings. If this condition does not hold, the user feels envy towards other group members. In this context, g_{ef} denotes the users for whom m-envy-freeness holds [23].

$$fairness_{m-envy}(g) = \frac{|g_{ef}|}{|g|} \tag{3.17}$$

An example of the calculation of a fairness estimate following the m-envy-freeness criteria ($m = 1$ and $x = 25\%$) is given in Table 3.11. In this example, the g_{ef} value is 3 since u_1, u_3, and u_4 are group members of group g_1 with at least one item each for which they are in the top 25% of the item ratings.

g_1	item ratings			m-envy
	t_1	t_2	t_3	
u_1	5.0	4.0	1.0	1
u_2	4.0	3.0	2.0	0
u_3	4.0	5.0	5.0	1
u_4	3.0	3.0	5.0	1
$fairness_{m-envy}$	-	-	-	$\frac{3}{4}$

Table 3.11: Fairness based on m-envy-freeness (m-envy) where $m = 1$ and $x = 25\%$.

The same approach to evaluate the fairness of recommendations proposed by a group recommender system can be applied in the context of *content-based*, *constraint-based*, and *critiquing-based recommendation*.

3.7 Conclusions and Research Issues

With a specific focus on group recommender systems, we have provided an overview of evaluation techniques. We have learned that the evaluation of group recommender systems can often be accomplished by employing standard evaluation approaches from single user recommender systems [24]. We want to emphasize that there are several other metrics that one might want to consider when evaluating group recommender systems. Examples thereof are *trust*, *privacy*, and *performance*. Importantly, the usefulness of some evaluation metrics also depends on the item domain. For example, food recommender systems are not only following the goals of accuracy but also other criteria such as healthiness [22, 27]. An open research issue in the context of group recommender systems but also single-user recommender systems are evaluation metrics for complex, for example, configurable items. In this chapter, we have provided a couple of examples of metrics for complex items, but a more in-depth analysis and provision of corresponding metrics is an issue for future research. Synthesis approaches to generate groups sound like a promising and 'cheap' alternative to studies with real groups. Often, clustering approaches are applied to derive groups from single user datasets – see, for example, Baltrunas et al. [2]. Group synthesis can also be based on analyzing social networks where social ties can serve as an indicator for group membership [18]. However, it should be mentioned that such approaches are based on simulations and should not replace controlled lab-studies, crowd sourcing studies, or naturalistic online tests. For an overview of datasets related to single user recommender systems and existing software frameworks that can serve as a basis for developing group recommender systems, we refer to Said et al. [21].

References

1. M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. Motahari-Nezhad, E. Bertino, and S. Dustdar. Quality Control in Crowdsourcing Systems: Issues and Directions. *IEEE Internet Computing*, 17(12):76–81, 2013.
2. L. Baltrunas, T. Makcinskas, and F. Ricci. Group Recommendations with Rank Aggregation and Collaborative Filtering. In *4th ACM Conference on Recommender Systems*, pages 119–126, Barcelona, Spain, 2010.
3. R. Burke. Multisided Fairness for Recommendation. In *2017 Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017)*, pages 1–5, 2017.
4. P. Campos, F. Díez, and I. Cantador. Time-aware Recommender Systems: A Comprehensive Survey and Analysis of Existing Evaluation Protocols. *User Modeling and User-Adapted Interaction*, 24(1–2):67–119, 2014.
5. J. Castro, F. Quesada, I. Palomares, and L. Martínez. A Consensus-Driven Group Recommender System. *Intelligent Systems*, 30(8):887–906, 2015.
6. S. Chang, F. Harper, L. He, and L. Terveen. CrowdLens: Experimenting with Crowd-Powered Recommendation and Explanation. In *10th International AAAI Conference on Web and Social Media (ICWSM'16)*, pages 52–61. AAAI, 2016.
7. T. DePessemier, J. Dhondt, and L. Martens. Hybrid Group Recommendations for a Travel Service. *Multimedia Tools and Applications*, 76(2):2787–2811, 2017.
8. W. Dubitzky, M. Granzow, and D. Berrar. *Fundamentals of Data Mining in Genomics and Proteomics*. Springer, 2007.
9. A. Felfernig, M. Atas, T.N. Trang Tran, and M. Stettinger. Towards Group-based Configuration. In *International Workshop on Configuration 2016 (ConfWS'16)*, pages 69–72, 2016.
10. A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-based Configuration: From Research to Business Cases*. Elsevier/Morgan Kaufmann Publishers, 1st edition, 2014.
11. A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, S. Reiterer, and M. Stettinger. Basic Approaches in Recommendation Systems. *Recommendation Systems in Software Engineering*, pages 15–37, 2013.
12. A. Felfernig, M. Schubert, and C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, 26(1):53–62, 2012.

13. M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *4th ACM Conference on Recommender Systems*, pages 257–260, Barcelona, Spain, 2010.
14. F. Harper and J. Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2015.
15. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
16. B. Knijnenburg and M. Willemsen. Evaluating Recommender Systems with User Experiments. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 309–352. Springer, 2015.
17. J. Masthoff and A. Gatt. In Pursuit of Satisfaction and the Prevention of Embarrassment: Affective State in Group Recommender Systems. *User Modeling and User-Adapted Interaction (UMUAI)*, 16(3–4):281–319, 2006.
18. A. Mislove, B. Viswanath, K. Gummadi, and P. Druschel. You are Who you Know: Inferring User Profiles in Online Social Networks. In *3rd ACM Conference on Web Search and Data Mining (WSDM'10)*, pages 251–260, New York, USA, 2010.
19. D. Parra and S. Sahebi. Recommender Systems: Sources of Knowledge and Evaluation Metrics. In *Advanced Techniques in Web Intelligence-2: Web User Browsing Behaviour and Preference Analysis*, pages 149–175. Springer, 2013.
20. L. Quijano-Sanchez, J. Recio-García, B. Díaz-Agudo, and G. Jiménez-Díaz. HAPPY MOVIE: A Group Recommender Application in Facebook. In *24th International Florida Artificial Intelligence Research Society Conference*, pages 419–420, Palm Beach, FL, USA, 2011.
21. A. Said, D. Tikk, and P. Cremonesi. Benchmarking – A Methodology for Ensuring the Relative Quality of Recommendation Systems in Software Engineering. In *Recommender Systems in Software Engineering*, pages 275–300, 2014.
22. H. Schaefer, S. Hors-Fraile, R. Karumur, A. Valdez, A. Said, H. Torkamaan, H. Ulmer, and C. Trattner. Towards health (aware) Recommender Systems. In *DH 2017*, 2017.
23. D. Serbos, S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas. Fairness in Package-to-Group Recommendations. In *WWW'17*, pages 371–379. ACM, 2017.
24. G. Shani and A. Gunawardana. Evaluating Recommender Systems. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer, 2011.
25. M. Stettinger, A. Felfernig, G. Leitner, and S. Reiterer. Counteracting Anchoring Effects in Group Decision Making. In *23rd Conference on User Modeling, Adaptation, and Personalization (UMAP'15)*, volume 9146 of *LNCS*, pages 118–130, Dublin, Ireland, 2015.
26. J. Tiihonen and A. Felfernig. Towards Recommending Configurable Offerings. *International Journal of Mass Customization*, 3(4):389–406, 2010.
27. C. Trattner and D. Elsweiler. Investigating the Healthiness of Internet-sourced Recipes: Implications for Mean Planning and Recommender Systems. In *26th International Conference on the World Wide Web*, pages 489–498, 2017.
28. T. Ulz, M. Schwarz, A. Felfernig, S. Haas, A. Shehadeh, S. Reiterer, and M. Stettinger. Human Computation for Constraint-based Recommenders. *Journal of Intelligent Information Systems (JIIS)*, 49(1):37–57, 2017.
29. L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *ACM Conference on Recommender Systems (RecSys'17)*, pages 107–115. ACM, 2017.
30. A. Zapata, V. Menéndez, M. Prieto, and C. Romero. Evaluation and Selection of Group Recommendation Strategies for Collaborative Searching of Learning Objects. *Int. Journal of Human Computer Studies*, 76:22–39, 2015.