Chapter 1
Decision Tasks and Basic Algorithms

Alexander Felfernig, Müslüm Atas, and Martin Stettinger[a][b]

[a] Citation: Alexander Felfernig, Müslüm Atas, and Martin Stettinger. Decision Tasks and Basic Algorithms, in: Group Recommender Systems – An Introduction, Alexander Felfernig, Ludovico Boratto, Martin Stettinger, and Marko Tkalčič (eds.), Springer, pp. 3–26, ISBN: 978-3-319-75066-8, 2018.

[b] This is a pre-print version of the chapter published in the book "Group Recommender Systems: An Introduction": http://www.springer.com/us/book/9783319750668.

Abstract Recommender systems are *decision support systems* helping users to identify one or more items (solutions) that fit their wishes and needs. The most frequent application of recommender systems nowadays is to propose items to *individual users*. However, there are many scenarios where a *group of users* should receive a recommendation. For example, think of a group decision regarding the *next holiday destination* or a group decision regarding a *restaurant to visit for a joint dinner*. The goal of this book is to provide an introduction to *group recommender systems*, i.e., recommender systems that determine recommendations for groups. In this chapter, we provide an introduction to basic types of recommendation algorithms for individual users and characterize related decision tasks. This introduction serves as a basis for the introduction of group recommendation algorithms in Chapter 2.

1.1 Introduction

A recommender system is *a specific type of advise-giving or decision support system that guides users in a personalized way to interesting or useful objects in a large space of possible options or that produces such objects as output* [16, 19, 23, 29, 43, 66, 76, 81, 82]. A decision problem/task emerges if a person or a group of persons have an idea about a desired state [33]. If there are different options to achieve the desired state, the *decision task* to be solved is to identify items or actions that help to approach the target state in a suitable fashion. Recommender systems can provide help in such a context by trying to find the suitable items or actions that help to best reach the envisioned target. Arriving at a choice can be seen as the result of a collaboration between the user and the recommender system. Recommender systems support 'good' choices within reasonable time spans including corresponding justifications provided in terms of explanations [42, 80].

There are different ways in which a recommender can support users in decision making processes. First, it can act as a *supporter* to figure out *candidate items*, i.e., a large number of alternatives is reduced to a so-called *consideration set* – selecting the favorite option is left to the user. Second, the recommender can help the user select from the items in the consideration set, for example, by representing them in convenient ways and providing explanations of why they have been recommended. Only in 'extreme' cases is the *decision making authority taken over* by the recommender itself. Examples include music recommendation in a fitness studio, the recommendation of information units on a public display, and the automated adaptation of parameter settings such as light intensity in a smarthome [50].

An example of a single-user recommendation scenario is the following: when navigating an online sales platform with the goal to find a book related to the topic of, for example, *deep learning*, a recommender system will identify related books and propose these to the user of the online platform. In this scenario, the envisioned target state is to find a suitable book on the mentioned topic and the options to achieve this state are the different existing books on the topic of *deep learning*. Finding a book in an online sales platform is typically a *single user decision scenario*. However, there are also scenarios where a group of users has to make a decision. In this context, a recommender system must take into account the potentially conflicting preferences of different group members. Such a situation makes the recommendation task different and often more challenging.

The main focus of this book is *recommendation techniques that provide help in scenarios where a group of users is engaged*. In many scenarios, the presentation of recommendations to groups is a more natural approach than trying to address individual users [41, 56, 58]. For example, music recommendations in fitness studios have to take into account the preferences of all individuals currently present in the studio [59]. Stakeholders in a software project have to establish agreement regarding the requirements/features that have to be developed within the scope of the next release [64]. Personnel decisions are often taken in groups, i.e., a group has to decide which job applicant will be hired [78]. Groups of friends have to decide about the hotel for the next summer holidays or a skiing resort for the next winter holidays [39, 60]. A public display should be personalized in order to be able to display information to persons currently in the surrounding [41]. Finally, travel groups should receive a personalized museum guidance in such a way that the personal preferences of group members are fulfilled [28, 47].

1.2 Characteristics of Decision Tasks

Decision tasks differ with regard to various aspects [69]. In the following, we introduce *basic dimensions of decision tasks* (see [33]) which will help to better understand which decisions are supported by group recommenders (see Table 1.1).[1]

Complexity. We interpret complexity of decision tasks in terms of the *number of decision dimensions* and the *degree of item involvement* [33, 38]. Depending on the underlying item domain, humans will invest more or less time to come to a

[1] As mentioned in [33], this characterization of decision tasks is not complete but a good basis for discussing properties relevant for group recommenders.

dimension	characteristic
complexity	low .. high
structuredness	low .. high
decision type	choice .. design
sentiment	opportunity .. threat
dependence	yes .. no
level	original .. meta
actor	person .. group

Table 1.1: Characteristics of decision tasks [33].

decision, i.e., to achieve an acceptable trade-off between decision effort and accuracy [67]. Items with higher related decision efforts are often denoted as *high-involvement items* whereas items with less related decision efforts are denoted as *low-involvement items* [70]. Suboptimal decisions have a much higher negative impact in the context of high-involvement items. For example, when purchasing an *apartment*, a suboptimal decision manifests in search efforts for a new apartment, unnecessary payments, relocation costs, and additional time efforts [15]. In contrast, risks related to the purchasing of a low-quality book are negligible – in the worst case, the user will provide negative feedback on the book and try to find other alternatives that better fit his/her wishes and needs. When purchasing a book, the number of decision dimensions is low – examples of dimensions are *price* and *quality*. The number of decision dimensions of *apartments* is much higher (e.g., price, quality of public transport, neighborhood, schools in the neighborhood, etc.). Aspects that further increase decision complexity especially in group decision scenarios are, for example, *contradicting preferences of group members*, *personal relationships*, *personality factors*, and *emotion-related aspects* (see Chapter 9).

Structuredness. We interpret structuredness of decision tasks as the degree to which underlying processes and decision policies are defined. Decision tasks are often characterized by undefined processes and related decision policies are not pre-defined but developed and adapted in the course of the decision process. If a group of users has to decide on a holiday destination for the next summer, a recommender system can propose alternative destinations but it is unclear which of the alternatives will be chosen by the group. The final decision is something that has to be made by group members (or an individual user) and is in many cases not handled by the underlying decision support environment. There are exceptions of the rule, for example, music recommendations in fitness centers and information units shown on public displays. Specific decision types follow a *formalized process*. For example, electoral systems are defined by precise rules that determine how elections and referendums have to be conducted (the process) and how the results are determined (decision making).

Decision Type. Decision tasks are often defined on the basis of known alternatives or parameters out of which one or more alternatives (values) should be selected. If alternatives (parameters) are predefined, the underlying decision task can be regarded as a basic *choice task* [42]. Choice problems are considered as central application area for recommender systems [42]. The other extreme are so-called *design tasks*, where alternatives are not predefined but created throughout a decision process. Design tasks are often related to creative acts where persons develop ideas and solutions. In 'pure' design scenarios, the application of recommendation technologies is not widespread [75]. However, there are many scenarios located in-between basic choice tasks and 'pure' design tasks. For example, knowledge-based configuration [22] can be considered as a simpler type of design task where a solution is identified (configured) out of a set of pre-defined component types. In this context, the alternatives (parameter values) are known beforehand; due to the large option space, not all potential alternatives can be enumerated for performance reasons – billions of alternatives would have to be managed and the corresponding recommendation algorithms would become inefficient [13] (see Chapter 7).

Sentiment. Decision support with included recommendation support is very often *opportunity-related*, i.e., the goal is related to an opportunity and the best solution to achieve a goal should be identified. Examples thereof are purchasing a book to better understand a certain topic or choosing a holiday destination to spend unforgettable days somewhere abroad. A similar argument holds for item domains such as songs, digital equipment, food, and financial services. Certainly, decision problems also exist in contexts where alternative outcomes can be considered as negative ones. For example, choosing between alternative options to liquidate a company – in this scenario, every outcome can be considered as a negative one (the company gets liquidated). However, recommender systems can help to minimize damage, for example, on the basis of a structured utility analysis [79].

Dependence. We regard decision tasks as dependent if the outcome of a decision has an impact on another decision. For example, the purchase of a movie typically does not require a follow-up decision regarding the purchase of the next movie or different item. Dependent decision tasks occur when one decision at an earlier point of time leads to follow-up decision tasks at a later point of time. An example of such a decision task is *requirements release planning* where for each software requirement it has to be decided when the requirement should be implemented [64]. Consequently, decisions taken in early phases of a software project can have an impact on or trigger decisions later in the project. For example, a decision that a requirement should be implemented could trigger decisions regarding additional resources in order to be able to provide the promised software functionalities in time.

Level. We can differentiate between *original decisions* operating on the object level and decisions on the *meta-level* [33]. The first type is omni-present in many recommendation-supported scenarios – the underlying task is to identify and choose items of relevance. In contrast, meta-decisions are decisions about the qualities of a decision process and the way decisions are taken. For example, a group could decide to use *majority voting* when it comes to the election of the next chairman. A meta-decision in this context is to decide about the election formalism – related alternatives can be, for example, *relative majority and a single-shot election* or *absolute majority in a potentially multi-level election process* [51]. In many decision scenarios – especially in the context of group decision making – recommendations have an advisory function but are not considered imperative.

Actor. Many recommendation approaches support individual decision making where recommendation algorithms are focusing on determining recommendations for individual users. The focus of this book are recommender systems that support *decision making for groups of users*. The following types of groups are introduced in [7]: (1) *established groups* with shared and longterm common interests (e.g., *conference committees* taking decisions about conference venues or *families* taking a decision about purchasing a house), (2) *occasional groups* with a common aim in a particular moment (e.g., a group of persons jointly participating in a museum tour), (3) *random groups* (e.g., persons in a fitness center or persons around a public display), and (4) *automatically identified groups* where individuals with similar preferences have to be grouped (e.g., distribution of seminar papers to students and distribution of conference papers to reviewers).

1.3 Recommendation Algorithms for Individual Users

Recommender systems [43, 58] propose items of potential interest to an individual user or a group of users.[2] They are applied in item domains such as books [52], web sites [68], financial services [18], and software artifacts [24, 25]. In the following, we introduce *collaborative filtering* [31, 48], *content-based filtering* [68], *constraint-based* [8, 16], *critiquing-based* [10, 11], and *hybrid recommendation* [9] which are basic recommendation approaches. The items in Table 1.2 (travel destinations) serve as examples to demonstrate how basic recommendation algorithms operate. In Chapter 2, we show how these approaches can be integrated into corresponding group recommendation scenarios.

travel destination (item)	name	beach	city tours	nature	entertainment
t_1	Vienna		x		x
t_2	Yellowstone			x	
t_3	New York	x	x		x
t_4	Blue Mountains			x	
t_5	London		x		x
t_6	Beijing		x		x
t_7	Cape Town	x	x	x	x
t_8	Yosemity			x	
t_9	Paris		x		x
t_{10}	Pittsburgh		x		x

Table 1.2: Example set of travel destinations. These items will be used across the following sections for demonstration purposes. Each travel destination is described by a set of meta-characteristics (categories), for example, travel destination *Yellowstone* is famous for its experience of *nature*.

[2] Parts of this section are based on a discussion of recommendation technologies given in [24].

Collaborative Filtering

Collaborative filtering is based on the idea of word-of-mouth promotion where opinions of relatives and friends play a major role when taking a decision [12, 48, 52]. In online scenarios, family members and friends are replaced by *nearest neighbors* who are users with preferences similar to the ones of the current user. In collaborative filtering, a *user* $\times$ *item* rating specifies to which extent a user likes an item. *Rating predictions* are determined by a recommender algorithm to estimate the extent a user will like an item he/she did not consume/evaluate up to now. A collaborative filtering recommender first determines k nearest neighbors ($k-NN$).[3] The preferences of nearest neighbors are used to extrapolate future item ratings of the current user. A $user \times item$ rating matrix that will be used in the following for explaining collaborative filtering is shown in Table 1.3. In this example, all users u_i visited travel destinations and provided a corresponding rating. In collaborative filtering, $user \times item$ ratings serve as input for the recommender.

item	name	u_1	u_2	u_3	u_4	u_a
t_1	Vienna	5.0			4.0	
t_2	Yellowstone	4.0				
t_3	New York		3.0	4.0	3.0	
t_4	Blue Mountains		5.0	5.0		4.0
t_5	London			3.0		
t_6	Beijing		4.5	4.0		4.0
t_7	Cape Town	4.0				
t_8	Yosemity		2.0			
t_9	Paris				3.0	
t_{10}	Pittsburgh				5.0	3.0
	average rating ($\overline{r_x}$)	4.33	3.625	4.0	3.75	3.67

Table 1.3: Example collaborative filtering rating matrix: travel destinations (items) t_i and ratings (we assume a rating scale of 1..5).

Collaborative filtering identifies the k-nearest neighbors of the current user u_a (see Formula 1.1)[4] and – based on the nearest neighbors – calculates a prediction for the current user's rating. When applying Formula 1.1, user u_2 is identified as the nearest neighbor of user u_a (see also Table 1.3). The similarity between u_a and another user u_x can be determined, for example, using the Pearson correlation coefficient [43] (see Formula 1.1) where TD_c is the set of items that have been rated by both users (u_a and u_x), r_{x,t_i} is the rating of user x for item t_i, and $\overline{r_x}$ is the average rating of user x. Similarity values resulting from Formula 1.1 can take values on a scale of [-1 .. +1]. Sometimes, 'neighbor' users with low or negative correlations with the current user are filtered out [1].

[3] We focus on *user-based collaborative filtering* which is a memory-based approach that – in contrast to model-based ones – operates on an uncompressed version of a user/item matrix [6, 12].

[4] We assume k=2 in our example.

$$similarity(u_a, u_x) = \frac{\sum_{t_i \in TD_c} (r_{a,t_i} - \overline{r_a}) \times (r_{x,t_i} - \overline{r_x})}{\sqrt{\sum_{t_i \in TD_c} (r_{a,t_i} - \overline{r_a})^2} \times \sqrt{\sum_{t_i \in TD_c} (r_{x,t_i} - \overline{r_x})^2}} \tag{1.1}$$

The similarity values for u_a calculated based on Formula 1.1 are shown in Table 1.4. For the purpose of our example, we assume the existence of at least two items per user pair (u_i, u_j) (i $\neq$ j) in order to be able to determine a similarity. This criterion holds for users u_2 and u_3.

	u_1	u_2	u_3	u_4
u_a	-	0.97	0.70	-

Table 1.4: Similarity between user u_a and the users $u_j \neq u_a$ determined based on Formula 1.1. If the number of commonly rated items is below 2, no similarity between the two users is calculated.

A challenge when determining the similarity between users is the *sparsity* of the rating matrix. Users typically provide ratings for only a very small subset of the offered items. For example, given a large movie dataset that contains thousands of entries, a user will typically be able to rate only a few dozen. One approach to this problem is to take into account the number of commonly rated items as a *correlation significance* [37], i.e., the higher the number of commonly rated items, the higher is the significance of the corresponding correlation. For further information regarding the handling of sparsity we refer to [37, 43].

The information about the set of users with a rating behavior similar to that of the current user (nearest neighbors NN) is the basis for predicting the rating of user u_a for an *item t* that has so far not been rated by u_a (see Formula 1.2).

$$prediction(u_a, t) = \hat{r}(u_a, t) = \overline{r_a} + \frac{\sum_{u_j \in NN} similarity(u_a, u_j) \times (r_{j,t} - \overline{r_j})}{\sum_{u_j \in NN} similarity(u_a, u_j)} \tag{1.2}$$

Based on the ratings of the nearest neighbors of u_a, we are able to determine a prediction for u_a (see Table 1.5). The nearest neighbors of u_a are assumed to be u_2 and u_3 (see Table 1.4). The travel destinations rated by the nearest neighbors but not rated by u_a are t_3, t_5, and t_8. Due to the determined predictions (Formula 1.2), item t_3 would be ranked higher than the items t_5 and t_8 in a recommendation list. For a discussion of advanced collaborative recommendation approaches we refer the reader to [49, 74].

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
u_2	-	-	3.0	5.0	-	4.5	-	2.0	-	-
u_3	-	-	4.0	5.0	3.0	4.0	-	-	-	-
u_a	-	-	-	4.0	-	4.0	-		-	3.0
prediction for u_a	-	-	3.30 $\surd$	-	2.66	-	-	2.04	-	-

Table 1.5: Collaborative filtering based recommendations (predictions) for items that have not been rated by user u_a up to now.

Content-based Filtering

This approach is based on the assumption of monotonic personal interests [68]. For example, users interested in *political news* are typically not changing their interest profile from one day to another. On the contrary, they will also be interested in the topic in the (near) future. In online scenarios, content-based recommenders are applied, for example, when it comes to the recommendation of websites [68].

Content-based filtering is based on (a) a set of users and (b) a set of categories (or keywords) that have been assigned to (or extracted from) the set of items. It compares the content of already consumed items with new items, i.e., it finds items that are similar to those already consumed (positively rated) by the user (u_a). The basis for determining such a similarity are *keywords* extracted from the item content descriptions (e.g., keywords extracted from news articles) or *categories* if items have been annotated with the relevant meta-information. Readers interested in the principles of keyword extraction are referred to [43]. In this book, we focus on content-based recommendation which exploits item categories (see Table 1.2).

Content-based filtering will now be explained using Tables 1.2, 1.6, and 1.7. Table 1.2 provides an overview of the relevant items and the assignments of items to categories. Table 1.6 provides information on which categories are of relevance for our example users. For example, user u_1 is interested in items related to all categories. Since user u_a rated the items t_4, t_6, and t_{10} (see Table 1.3), we can infer that u_a is interested in the categories *City Tours*, *Nature*, and *Entertainment* (see Table 1.6) where items related to the categories *City Tours* and *Entertainment* have been evaluated twice and items related to *Nature* have been evaluated once by u_a.

category	u_1	u_2	u_3	u_4	u_a
Beach	x	x	x	x	-
City Tours	x	x	x	x	x
Nature	x	x	x	-	x
Entertainment	x	x	x	x	x

Table 1.6: Example category interests of users: users $u_1 - u_3$ visited travel destinations that cover all available categories. If a user accessed an item at least once with a rating ≥ 3 (see Table 1.3), it is inferred that the user is interested in this item.

item	rating (u_a)	name	beach	city tours	nature	entertainment	similarity(u_a, t_i)
t_1		Vienna	-	x	-	x	$\frac{4}{5}$
t_2		Yellowstone	-	-	x	-	$\frac{1}{2}$
t_3		New York	x	x	-	x	$\frac{2}{3}$
t_4	4.0	Blue Mountains	-	-	x	-	–
t_5		London	-	x	-	x	$\frac{4}{5}$
t_6	4.0	Beijing	-	x	-	x	–
t_7		Cape Town	x	x	x	x	$\frac{6}{7}\surd$
t_8		Yosemity	-	-	x	-	$\frac{1}{2}$
t_9		Paris	-	x	-	x	$\frac{4}{5}$
t_{10}	3.0	Pittsburgh	-	x	-	x	–
user u_a				x	x	x	

Table 1.7: Example of content-based filtering: *user* u_a has already consumed the items t_4, t_6, and t_{10} (see Table 1.3). The item most similar (see Formula 1.3) to the preferences of u_a is t_7.

If we are interested in an item recommendation for the user u_a, we have to search for those items which are most similar to the items that have already been consumed (evaluated) by u_a. This relies on the simple similarity metric shown in Formula 1.3 (Dice coefficient which is a variation of the Jaccard coefficient 'intensively' taking into account category commonalities – see also [43]). The major difference to the similarity metrics introduced in the context of collaborative filtering is that similarity is measured using categories (in contrast to ratings).

$$similarity(u_a, item) = \frac{2 * |categories(u_a) \cap categories(item)|}{|categories(u_a)| + |categories(item)|} \tag{1.3}$$

Constraint-based Recommendation

Compared to the approaches of collaborative filtering and content-based filtering, *constraint-based recommendation* [18, 16, 63] does not primarily rely on item ratings and textual item descriptions but on deep knowledge about the offered items. Such deep knowledge (semantic knowledge [18]) describes an item in more detail and thus allows for a different recommendation approach (see Table 1.8).

Constraint-based recommendation relies on (a) a set of rules (constraints) and (b) a set of items. Depending on the user requirements (a set of search criteria), rules (constraints) describe which items should be recommended. The current user (u_a) articulates his/her requirements (preferences) in terms of item property specifications which are internally represented as rules (constraints). In our example, constraints are represented solely by user requirements, no further constraint types are included. An example of a constraint is the following: *topics* = *city tours* (the user is primarily interested in travel destinations allowing city tours). For a detailed discussion of further constraint types we refer the reader to [18]. Constraints are interpreted and the resulting items are presented to the user. A detailed discussion

item	name	season	topics	eval
t_1	Vienna	1110	city tours, entertainment	4.5
t_2	Yellowstone	1110	nature	4.0
t_3	New York	1011	city tours, entertainment	3.3
t_4	Blue Mountains	1001	nature	5.0
t_5	London	1010	city tours, entertainment	3.0
t_6	Beijing	1010	city tours, entertainment	4.7
t_7	Cape Town	1111	beach, city tours, nature, entertainment	4.0
t_8	Yosemity	1110	nature	2.0
t_9	Paris	1011	city tours, entertainment	3.0
t_{10}	Pittsburgh	1010	city tours	5.0

Table 1.8: Slightly adapted travel destinations described based on *season* (digit 1 indicates a recommended season and 0 indicates a non-recommended one; seasons start with *spring*), associated *topics*, and average user rating (*eval*).

of reasoning mechanisms that are used in constraint-based recommendation can be found in [16, 20, 26]. In order to determine a recommendation in a constraint-based recommendation scenario, a *recommendation task* has to be solved.

Definition (Recommendation Task). A recommendation task can be defined by the tuple (R,I) where R represents a set of user requirements and I represents a set of items (in our case: travel destinations $t_i \in I$). The goal is to identify those items in I which fulfill the given user requirements (preferences).

A solution for a recommendation task (also denoted as recommendation) can be defined as follows.

Definition (Solution for a Recommendation Task). A solution for a recommendation task (R,I) is a set $S \subseteq I$ such that $\forall t_i \in S : t_i \in \sigma_{(R)} I$ where σ is the selection operator of a conjunctive query [20], R represents a set of selection criteria (represented as constraints), and I represents an item table (see, e.g., Table 1.8). If we want to restrict the set of item properties shown to the user in a result set (recommendation), we have to additionally include projection criteria π as follows: $\pi_{(attributes(I))}(\sigma_{(R)} I)$.

In our example, we show how to determine a solution for a given recommendation task based on a conjunctive query where user requirements are used as selection criteria (constraints) on an item table I. If we assume that the user requirements are represented by the set $R = \{r_1 : season = winter, r_2 : topics = city\ tours\}$ and the item table I consists of the elements shown in Table 1.8 then $\pi_{(item)}(\sigma_{(season=winter \wedge topics=city\ tours)} I) = \{t_3, t_7, t_9\}$, i.e., these three items are consistent with the given set of requirements.

Ranking Items. Up to now we know which items can be recommended to a user. One widespread approach to rank items is to define a utility scheme which serves as a basis for the application of *Multi Attribute Utility Theory* (MAUT).[5] Items can be evaluated and ranked with respect to a set of *interest dimensions*. In travel destinations, example interest dimensions are *security* (security level of the travel destination), *attractiveness* (estimated attractiveness of the travel destination), and *crowd-*

[5] A detailed discussion of MAUT in constraint-based recommendation is given in [2, 18, 21].

edness (degree of crowdedness of the travel destination). The first step to establish a MAUT scheme is to relate the interest dimensions with the given set of items. An example thereof is shown in Table 1.9 where the city of *Vienna* receives the highest evaluations with regard to the dimensions *security* and *attractiveness* and a low evaluation with regard to *crowdedness*.

item	name	security	attractiveness	crowdedness
t_1	Vienna	5	5	2
t_2	Yellowstone	4	4	4
t_3	New York	3	5	1
t_4	Blue Mountains	4	3	5
t_5	London	3	4	1
t_6	Beijing	3	3	1
t_7	Cape Town	2	3	3
t_8	Yosemity	4	4	4
t_9	Paris	3	5	1
t_{10}	Pittsburgh	3	3	3

Table 1.9: Travel destinations described with regard to the dimensions *security* (high evaluation represents a high security), *attractiveness* (high evaluation represents a high attractiveness), and *crowdedness* (high evaluation represents a low crowdedness). For example, *security = 5* for the item *Vienna* indicates the highest possible *contribution* to the dimension *security* (scale 1..5).

We are now able to determine the user-specific utility of each individual item. The calculation of *item* utilities for a specific user u_a can be based on Formula 1.4.

$$utility(u_a, item) = \sum_{d \in Dimensions} contribution(item, d) \times weight(u_a, d) \tag{1.4}$$

If we assume that the current user u_a assigns a weight of 0.6 to the dimension *security* (weight(u_a,security)=0.6), a weight of 0.3 to the dimension *attractiveness* (weight(u_a,attractiveness)=0.3), and a weight of 0.1 to the dimension *crowdedness* (weight(u_a,crowdedness)=0.1), then the user-specific utilities of the individual items (t_i) are the ones shown in Table 1.10.

Managing Inconsistencies. In constraint-based recommendation scenarios, we have to deal with situations where no solution (recommendation) can be identified for a given set of user requirements, i.e., $\sigma_{(R)}I = \emptyset$. In such situations, we are interested in proposals for requirements changes such that a solution can be found. For example, if a user is interested in travel destinations for *entertainment* with an overall *evaluation of 5.0* in the summer season, then no solution can be provided for the given set of requirements $R = \{r_1 : season = summer, r_2 : topics = entertainment, r_3 : eval = 5.0\}$.

item	security	attractiveness	crowdedness	utility
t_1	3.0	1.5	0.2	4.7 $\surd$
t_2	2.4	1.2	0.4	4.0
t_3	1.8	1.5	0.1	3.4
t_4	2.4	0.9	0.5	3.8
t_5	1.8	1.2	0.1	3.1
t_6	1.8	0.9	0.1	2.8
t_7	1.2	0.9	0.3	2.4
t_8	2.4	1.2	0.4	4.0
t_9	1.8	1.5	0.1	3.4
t_{10}	1.8	0.9	0.3	3.0

Table 1.10: Item-specific *utility* for user u_a (*utility*(u_a,t_i)) assuming the personal preferences {weight(u_a,security)=0.6, weight(u_a,attractiveness)=0.3, weight(u_a,crowdedness)=0.1}, item t_1 has the highest utility for user u_a.

User support in such situations can be based on the concepts of conflict detection [44] and model-based diagnosis [13, 17, 72]. A conflict (or conflict set) with regard to an item set I in a given set of requirements R can be defined as follows.

Definition (Conflict Set). A conflict set is a set $CS \subseteq R$ such that $\sigma_{(CS)}I = \emptyset$. CS is minimal if there does not exist a conflict set CS' with $CS' \subset CS$.

In our example, we are able to determine the following minimal conflict sets CS_i: $CS_1 : \{r_1, r_3\}$, $CS_2 : \{r_2, r_3\}$. We will not discuss algorithms that support the determination of minimal conflict sets but refer the reader to the work of Junker [44] who introduces a divide-and-conquer based algorithm with a logarithmic complexity in terms of the needed number of consistency checks.

Based on the identified minimal conflict sets, we are able to determine the corresponding (minimal) diagnoses. A diagnosis for a given set of requirements which is inconsistent with the underlying item table can be defined as follows.

Definition (Diagnosis). A diagnosis for a set of requirements $R = \{r_1, r_2, ..., r_n\}$ is a set $\Delta \subseteq R$ such that $\sigma_{(R-\Delta)}I \neq \emptyset$. A diagnosis Δ is minimal if there does not exist a diagnosis Δ' with $\Delta' \subset \Delta$.

In other words, a diagnosis (hitting set) is a minimal set of requirements that have to be deleted from R such that a solution can be found for R - Δ. The determination of the *complete set of minimal diagnoses* for a set of requirements inconsistent with the underlying item table (the corresponding conjunctive query results in $\emptyset$) is based on the construction of hitting set trees [72].

There are two possibilities of resolving the conflict set CS_1. If we decide to delete the requirement r_3, $\sigma_{(\{r_1,r_2\})}I \neq \emptyset$, i.e., a diagnosis has been identified ($\Delta_1 = \{r_3\}$) and – as a consequence – all CS_i have been resolved. Choosing the other alternative and resolving CS_1 by deleting r_1 does not result in a diagnosis since the conflict CS_2 is not resolved. Resolving CS_2 by deleting r_3 does not result in a minimal diagnosis, since r_3 already represents a diagnosis. The second (and last) minimal diagnosis that can be identified in our running example is $\Delta_2 = \{r_1, r_2\}$. For a detailed discussion of the underlying algorithm and analysis we refer to [27, 72]. Note that a diagnosis provides a hint to which requirements have to be changed. For a discussion of how requirement repairs (changes) are calculated, we refer to Felfernig et al. [20].

Critiquing-based Recommendation

Critiquing-based recommender systems support the navigation in the item space where in each critiquing cycle a reference item is presented to the user and the user either accepts the (recommended) item or searches for different solutions by specifying *critiques* (see Figure 1.1). Critiquing-based recommender systems are useful in situations where users are not experts in the item domain and prefer to specify their requirements on the level of critiques [46]. Critiques are basic criteria that are used for determining new recommendations which take into account the (changed) preferences of the current user. Examples of such critiques in the context of our running example are *higher level of security* and *higher attractiveness*. Critiques are used as search criteria to identify corresponding candidate items, i.e., items that are shown to the user if he/she has specified critiques on the current reference item. Critiquing-based recommenders often search for items that are similar to the current reference item and additionally take into account the new criteria defined as critiques. When searching for a new reference item, similarity and diversity metrics are applied to systematically guide the navigation in the item space [43].

Similarity metrics in critiquing-based approaches are used to determine the similarity between a reference item currently shown to the user and a set of candidate items to be shown to the user in the next critiquing cycle. Example similarity metrics often used in the context of critiquing-based recommendation scenarios are represented by Formulae 1.5 – 1.9. In this context, $sim(r,c)$ denotes the similarity between the reference item r and the candidate item c, the subroutine $s(r.i,c.i)$ is represented by different *attribute-level similarity metrics* (MIB, LIB, NIB, and EIB). If a higher attribute value is better than a lower one, *More-Is-Better* (MIB) (Formula 1.6) is used to evaluate the attribute of a candidate item ($c.i$). Vice versa, if low attributes values are considered better, the *Less-Is-Better* (LIB) similarity metric is used (Formula 1.7). Furthermore, *Nearer-Is-Better* (NIB) (Formula 1.8) is used if the attribute value of the candidate item should be as near as possible to the attribute $r.i$. Finally, *Equal-Is-Better* (EIB) is used in situations where attribute values should be equal (Formula 1.9). Besides taking into account the similarity between the reference item and candidate items, some critiquing-based systems also take into account the *compatibility* of a candidate item with regard to the complete critiquing history [60]. Thus, a trade-off between similarity to the reference item and critique compatibility can be achieved.

$$sim(r,c) = \sum_{i \in attributes} \mathrm{s}(r.i,c.i) * w(i) \quad (\Sigma_{i \in attributes} w(i) = 1) \tag{1.5}$$

$$MIB : s(r.i,c.i) = \frac{val(c.i) - minval(r.i)}{maxval(r.i) - minval(r.i)} \tag{1.6}$$

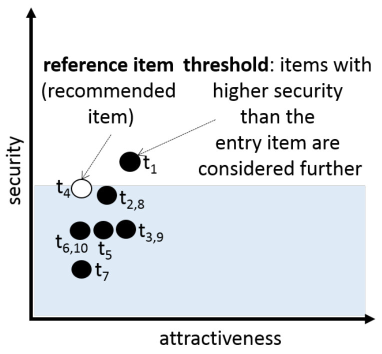
reference item (recommended item)
threshold: items with higher security than the entry item are considered further
security
attractiveness
t1
t4
t2,8
t3,9
t6,10
t5
t7

Fig. 1.1: Example of a critiquing scenario: an item (t_4) is shown as *reference item* to the user. The user specifies the critique 'higher security'. The new reference item is t_1 since it is consistent with the critique and the item most similar to t_4 (here, it is also the only remaining alternative).

$$LIB : s(r.i, c.i) = \frac{maxval(r.i) - val(c.i)}{maxval(r.i) - minval(r.i)} \tag{1.7}$$

$$NIB : s(r.i, c.i) = 1 - \frac{|val(r.i) - val(c.i)|}{maxval(r.i) - minval(r.i)} \tag{1.8}$$

$$EIB : s(r.i, c.i) = \begin{cases} 1 & \text{if } r.i = c.i \\ 0 & \text{otherwise} \end{cases} \tag{1.9}$$

If users are knowledgeable in the item domain, the application of search-based approaches such as constraint-based recommendation makes more sense. Different types of critiquing-based approaches primarily differ in terms of the way in which user preferences can be specified. *Unit critiquing* [10, 54] only supports the definition of critiques (change requests) that are related to a single item property (attribute). *Compound critiques* allow the specification of change requests over multiple item properties and thus allow to reduce the number of needed critiquing cycles [61]. Finally, *experience-based critiquing* takes into account critiquing histories of previous users to better predict reference items and thus to reduce the number of needed interaction cycles [54, 62]. For an in-depth discussion of different (additional) variants of critiquing-based recommendation we refer to [10, 11, 32, 54, 61, 73].

Hybrid Recommendation

After having discussed the basic recommendation approaches of *collaborative filtering*, *content-based filtering*, *constraint-based*, and *critiquing-based recommendation*, we will now present some possibilities to combine these.

A major motivation for the development of hybrid recommender systems is the opportunity to achieve a better accuracy [9]. There are different approaches to evaluate the accuracy of recommendation algorithms. These approaches can be categorized into *predictive accuracy metrics* such as the mean absolute error (MAE), *classification accuracy metrics* such as precision and recall, and *rank accuracy metrics* such as Kendall's Tau (see Chapter 3). For a discussion of accuracy metrics in recommendation scenarios for individual users we refer to Gunawardana and Shani [34] and Jannach et al. [43].

We now take a look at *example design types* of hybrid recommenders [9, 43]. These are *weighted*, *mixed*, and *cascade* (see Table 1.11). The basic assumption in the following example is that individual recommendation approaches return a list of *five* recommended items where each item has an assigned (recommender-individual) prediction *out of* $\{1.0, 2.0, 3.0, 4.0, 5.0\}$. For a more detailed discussion of hybridization strategies we refer the reader to Burke [9] and Jannach et al. [43].

method	description	calculation
weighted	predictions (s) of individual recommenders are summed up	score(item)=$\Sigma_{rec \in RECS}\ s(item, rec)$
mixed	recommender-individual predictions (s) are combined into one recommendation result	score(item) = zipper-function(item, RECS)
cascade	the prediction of one recommender $(n-1)$ is used as input for the next recommender (n)	score(item) = s(item, rec_n) $\leftarrow$ s(item,rec_{n-1}) $\leftarrow$ ...

Table 1.11: Examples of hybrid recommendation approaches (*RECS* = set of recommenders, *s* = recommender-individual prediction, *score* = item score).

Weighted. Weighted hybrid recommendation is based on the idea of deriving recommendations by combining the results (predictions) computed by individual recommenders. An example thereof is depicted in Table 1.12 where the individual item scores of a collaborative and a content-based recommender are summed up. Item t_8 receives the highest overall score (9.0) and is ranked highest by the weighted hybrid recommender.[6]

Mixed. Mixed hybrid recommendation is based on the idea that predictions of individual recommenders are shown in one integrated result. For example, the results of a collaborative filtering and a content-based recommender can be ranked as sketched in Table 1.13. Item scores can be determined, for example, on the basis of

[6] If two or more items have the same overall score, a possibility is to force a decision by lot; where needed, this approach can also be applied by other hybrid recommendation approaches.

items	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
s(t_i, collaborative filtering)	1.0	3.0	-	5.0	-	2.0	-	4.0	-	-
s(t_i, content-based filtering)	-	1.0	2.0	-	-	3.0	4.0	5.0	-	-
score(t_i)	1.0	4.0	2.0	5.0	0.0	5.0	4.0	9.0	0.0	0.0
ranking(t_i)	7	4	6	2	8	3	5	1	9	10

Table 1.12: Example of *weighted* hybrid recommendation: individual predictions are integrated into a score. Item t_8 receives the best overall score (9.0).

the zipper principle, i.e., the item with the highest collaborative filtering prediction value receives the highest overall score (10.0), the item with the best content-based filtering prediction value receives the second best overall score, and so forth.

items	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
s(l_i, collaborative filtering)	1.0	3.0	-	5.0	-	2.0	-	4.0	-	-
s(l_i, content-based filtering)	-	1.0	2.0	-	-	3.0	4.0	5.0	-	-
score(l_i)	4.0	8.0	5.0	10.0	0.0	6.0	7.0	9.0	0.0	0.0
ranking(l_i)	7	3	6	1	8	5	4	2	9	10

Table 1.13: Example of *mixed* hybrid recommendation: individual predictions are integrated into one score conform the zipper principle (best collaborative filtering prediction receives score=10.0, best content-based filtering prediction receives score=9.0 and so forth).

Cascade. The basic idea of cascade-based hybridization is that recommenders in a pipeline of recommenders exploit the recommendation of the upstream recommender as a basis for deriving their own recommendation. The constraint-based recommendation approach is an example of a cascade-based hybrid recommendation approach. First, items that are consistent with the given requirements are preselected by a conjunctive query Q. We can assume, for example, that s(item,Q) = 1.0 if the item has been selected and s(item,Q) = 0.0 if the item has not been selected. In our case, the set of requirements $\{r_1 : topics = nature\}$ in the running example leads to the selection of the items $\{t_2, t_4, t_7, t_8\}$. Thereafter, these items are ranked conform to their utility for the current user (utility-based ranking U). Based on the entries of Table 1.10, a utility-based ranking (U) would determine the item order utility(t_2) $\geq$ utility(t_8) $>$ utility(t_4) $>$ utility(t_7), assuming that the current user assigns a weight of 0.6 to the interest dimension *security* (weight(u_a,security) = 0.6), a weight of 0.3 to the interest dimension *attractiveness* (weight(u_a,attractiveness) = 0.3), and a weight of 0.1 to the interest dimension *crowdedness* (weight(u_a,crowdedness) = 0.1). In this example, the recommender Q is the first one and the results of Q are forwarded to the utility-based recommender.

Further Approaches. Further hybrid recommendation approaches are the following [9]. *Switching* denotes an approach where – depending on the current situation – a specific recommendation approach is chosen. For example, if a user has a low level of product knowledge, then a critiquing-based recommender will be chosen. Vice-versa, if the user is an expert, an interface will be provided where the user is enabled to explicitly state his/her preferences on a detailed level. *Feature combination* denotes an approach where different data sources are exploited by a single recommender. For example, a recommendation algorithm could exploit semantic item knowledge in combination with item ratings (see Table 1.8).

Outlook. Due to the increasing popularity of social platforms and online communities, group recommender systems are becoming an increasingly important technology [36, 58]. Example application domains of group recommendation technologies include tourism [60] (e.g., *which hotels or travel destinations should be visited by a group?*) and interactive television [57] (*which sequence of television programs will be accepted by a group?*). For the most part, group recommendation algorithms are operating on *simple items* such as hotels, travel destinations, and television programs. Examples of *complex items* are cars, round trips, and software release plans. In the remainder of this book, we will discuss different approaches that determine item recommendations for groups.

1.4 Relationship between Algorithms and Choice Patterns

As already mentioned, recommender systems can be regarded as important support for human decision making [42]. In the following, we will briefly discuss different patterns of choice, i.e., approaches people use to solve a decision task/problem. Following the concepts introduced in [40, 42], we will explain these patterns and point out related recommendation approaches. Our major motivation for this discussion is the increasingly perceived relevance of human decision making aspects in recommendation contexts [42, 55].

Socially-influenced Choice. If someone is interested in purchasing a digital camera, has no experiences in the item domain, but knows friends who are photography enthusiasts, there is a high probability that the opinion of these friends can have an impact on the camera purchase decision. People are influenced by the opinions and advice of friends beyond item recommendation, for example, in terms of social expectations of what is 'cool' or what is 'politically correct' [42]. From the viewpoint of recommender systems, *collaborative filtering* [48] simulates socially-influenced decisions where the preferences of nearest neighbors (social environment of the user) are taken as input for recommendations proposed to the current user. In such recommendation scenarios, *trust* plays an important role since only the preferences of trusted users should be taken into account by the recommendation algorithm [30, 65]. If available, an important aspect to be taken into account are *social networks* which can serve as a basis for representing a user's trust networks and also to determine the nearest neighbors relevant in the recommendation context [35]. In scenarios where groups of users have to make a decision, the personal opinion of a group member is in many cases influenced by his/her surroundings, i.e., other group members or even other groups [71]. Consequently, a key feature of group decisions is that social influence occurs not only in the environment but also within the group of decision makers themselves (see Chapter 9) - a fact that makes the socially-based pattern even more important than it is for individual choices.

Attribute-based Choice. This pattern is based on the idea that individual alternatives (items) are described by attributes which can be associated with importance values that reflect individual relevance. People can in principle evaluate an item by considering the values and importances of its attributes and computing something like a weighted sum for each item. More typically, they apply less effortful strategies. For example, *attributes* can serve as criteria to figure out whether an alternative should be taken into account (also denoted as *elimination by aspects* where users are kept in the loop), serve as a basis for the ranking of the available alternatives (utility-based ranking), or can be used for both purposes [5, 42, 67]. Due to the fact that importance of attributes and preferences for particular attribute levels are not stable and are not known in detail from the beginning [4], decision processes are iterative where users change or slightly adapt their preferences. Recommenders often integrated in such scenarios are *utility-based* and *constraint-based recommenders* [8, 16]. Constraint-based recommendation [16] is regarded as a special type of knowledge-based recommendation approach where the recommendation knowledge is represented in terms of explicitly defined attributes and constraints. The second type of knowledge-based recommendation (if we interpret item attributes and corresponding critiques as semantic knowledge) is represented by critiquing-based systems [11] where users specify change requests with regard to a reference alternative and the recommender system proposes a new candidate alternative that takes into account previous critiques. Critiques are typically defined on item attributes, therefore critiquing-based approaches can also be helpful to support attribute-based decision processes. There are situations where attribute-based decisions have to be supported for groups. Group members have to develop consensus with regard to a set of attributes, for example, arriving at a common set of importance weights or agreeing on the selection of an attribute value. If contradicting preferences occur, visualization and analysis methods can be applied to resolve inconsistencies [14].

Trial & Error based Choice. This choice pattern is often used in situations where the item domain is unknown and users want to better understand the domain and analyze the pros and cons of the different existing options in more detail [40]. For example, customers purchase the trial version of a software, test different cars, or evaluate some potential holidays within the scope of a short-term visit. In the context of recommender systems, trial & error based decisions are supported, for example, by critiquing-based recommendation approaches [11] where a user analyzes individual recommendations and then defines criteria that should additionally be taken into account when presenting the next item. Critiquing-based approaches support explorative navigation in large search spaces which better helps to develop an understanding of the item domain [43]. Critiquing-based recommendation also plays a role when it comes to determining recommendations for groups. For example, a group makes a decision with regard to a skiing resort to visit in the next winter season [60].

Experience-based Choice. This type of choosing can be applied to situations where experiences of users from the past directly influence current decisions, often making it unnecessary to apply any of the other choice patterns (see, e.g., [3]). For example, if someone purchased a book from a specific author or a specific publisher, this experience can be exploited in future purchasing scenarios and books from one consideration set could be preferred based on these experience-based criteria. Another example is car purchasing: if a customer is completely satisfied with his/her car (specific brand), there is a good chance that he/she will stick with the brand when purchasing a new car. In this context, positive feelings from the past trigger positive feelings about specific alternatives in ongoing decision processes [42]. From the point of view of recommendation approaches, content-based recommendation implements experience-based decisions in the sense that positively-rated purchases in the past represent major criteria to recommend similar items in the future. For example, a user liked a book of a specific author, therefore a new book from the same author is recommended to the user. We want to point out that in the recommender systems community, critiquing-based recommendation [11] is *also* considered as specific type of case-based recommendation [53, 77] where items are regarded as cases and items similar to a given set of user requirements (the case description) are recommended. Since cases represent experiences from the past, critiquing-based recommendation can also be considered as representative of experience-based decisions. This becomes even clearer if we take a look at recent experience-based critiquing approaches where experiences from previous recommendation sessions are taken into account and users with similar critiquing histories receive similar recommendations [54, 62]. Here, the experience-based pattern is supplemented with the socially-based, since the experiences of other persons are being exploited as well.

Consequence-based Choice. An alternative decision strategy is to think about the potential consequences of choosing a specific alternative. In contrast to decision patterns such as attribute-based decisions and socially-influenced decisions, thinking about the consequences of making a specific decision (choosing a specific alternative) constitutes an additional dimension of a decision process. One challenge in this context is the uncertainty about the consequences of taking a decision and how to best deal with this. Since anticipating and evaluating consequences is typically an effortful process, people are more likely to do so in some domains than in others. Low involvement items, i.e., items with lower risk impacts related to suboptimal decisions (e.g., movies, restaurants, and songs) will result in less investment in the analysis of the impact of choosing such items. In contrast, high-involvement items, i.e., items with higher associated risks, will result in higher investment in the analysis of the consequences of choosing a specific alternative. We want to point out that consequence-based decision patterns are often orthogonal, i.e., they play a role in combination with each of the mentioned recommendation approaches. Like all of the choice patterns, the consequence-based pattern can be combined with other patterns. For example, a chooser may apply the attribute-based pattern to quickly form a consideration set and then apply the consequence-based pattern to the consideration set. An important aspect of decision making is the relatively strong emphasis on the potential negative consequences of a decision – this aspect is taken into account in *prospect theory* [45], an asymmetric utility function which postulates that losses have a higher negative evaluation compared to equal gains.

Policy-based Choice. In the initial phase of a decision process, it can be the case that decision makers also define their decision policy. For example, during the weekend, if a movie recommender proposes a thriller and a cartoon, a family may apply a general policy of first consuming the movie that is suitable for children. They don't need to think in each case about the justification for this policy (i.e., that children need to go to bed earlier). When deciding about the next restaurant for a Christmas party, a company could have predefined the (meta-level) decision policy of *majority voting*, i.e., the majority defines which restaurant will be chosen for the next Christmas party. If two books with equal average ratings and high-quality reviews are in a consideration set, the customer could purchase the book from a publisher he/she previously found satisfying. A book customer may (for any of various reasons) have acquired the policy of buying a particular type of book from a particular publisher. In that case, the customer does not even need to consider books from other publishers. As these examples suggest, the policy-based decision pattern can play a role in the context of each basic recommendation scenario (*collaborative*, *content-based*, *constraint-based*, and *critiquing-based*).

1.5 Book Overview

In this chapter, we introduced recommender systems as a basic technology to support different decision scenarios. We gave an overview of recommendation algorithms and showed their application in the context of a scenario from the travel domain. Finally, we discussed the relationship between basic patterns of human choice and related supportive recommendation approaches. *The remainder of this book is organized as follows*. In Chapter 2, we provide an overview of group recommendation algorithms with the goal of showing how basic recommendation algorithms (collaborative filtering, content-based filtering, constraint-based, and critiquing-based recommendation) can be tailored to group settings. In this context, we show the relationship of group recommenders to the algorithms discussed in Chapter 1. In Chapter 3, we sketch approaches to evaluate group recommender systems. An overview of existing applications of group recommendation technologies is provided in Chapter 4. In Chapter 5, we focus on a discussion of ways to elicit and manage user preferences. Chapter 6 deals with explanation approaches in the context of group recommendation. In Chapter 7, we introduce additional decision scenarios, for example, group-based configuration, group-based resource balancing, and group-based release planning. Chapter 8 analyzes the existence and ways to counteract decision biases that can occur in the context of group decision making. Chapter 9 focuses on the role of personality and emotion in group recommendation. Finally, the book is concluded with a summary and an outlook (see Chapter 10).

References

1. C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 2016.
2. L. Ardissono, A. Felfernig, G. Friedrich, A. Goy, D. Jannach, G. Petrone, R. Schäfer, and M. Zanker. A Framework for the Development of Personalized, Distributed Web-based Configuration Systems. *AI Magazine*, 24(3):93–108, 2003.
3. T. Betsch and S. Haberstroh. *The Routines of Decision Making*. Lawrence Erlbaum Associates, 2005.
4. J. Bettman, M. Luce, and J. Payne. Constructive Consumer Choice Processes. *Journal of Consumer Research*, 25(3):187–217, 1998.
5. S. Bhatia. Associations and the Accumulation of Preference. *Psychological Review*, 120(3):522–543, 2013.
6. D. Billsus and M. Pazzani. Learning Collaborative Information Filters. In *15th International Conference on Machine Learning (ICML'98)*, pages 46–54, 1998.
7. L. Boratto and S. Carta. State-of-the-Art in Group Recommendation and New Approaches for Automatic Identification of Groups. In *Information Retrieval and Mining in Distributed Environments*, volume 324 of *Studies in Computational Intelligence*, pages 1–20. 2011.
8. R. Burke. Knowledge-based Recommender Systems. *Encyclopedia of Library and Information Systems*, 69(32):180–200, 2000.
9. R. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction (UMUAI)*, 12(4):331–370, 2002.
10. R. Burke, K. Hammond, and B. Young. The FindMe Approach to Assisted Browsing. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):32–40, 1997.
11. L. Chen and P. Pu. Critiquing-based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction (UMUAI)*, 22(1–2):125–150, 2012.
12. M. Ekstrand, J. Riedl, and J. Konstan. Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.
13. A. Falkner, A. Felfernig, and A. Haag. Recommendation Technologies for Configurable Products. *AI Magazine*, 32(3):99–108, 2011.
14. 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.
15. A. Felfernig, M. Atas, T.N. Trang Tran, M. Stettinger, and S. Polat-Erdeniz. An Analysis of Group Recommendation Heuristics for High- and Low-Involvement Items. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2017)*, pages 335–344, Arras, France, 2017.
16. A. Felfernig and R. Burke. Constraint-based Recommender Systems: Technologies and Research Issues. In *ACM International Conference on Electronic Commerce (ICEC08)*, pages 17–26, Innsbruck, Austria, 2008.
17. A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner. Consistency-based Diagnosis of Configuration Knowledge Bases. *Artificial Intelligence*, 152(2):213–234, 2004.
18. A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An Integrated Environment for the Development of Knowledge-based Recommender Applications. *Intl. Journal of Electronic Commerce (IJEC)*, 11(2):11–34, 2006.
19. A. Felfernig, G. Friedrich, and L. Schmidt-Thieme. Guest Editors' Introduction: Recommender Systems. *IEEE Intelligent Systems*, 22(3):18–21, 2007.
20. A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan. Plausible Repairs for Inconsistent Requirements. In *IJCAI'09*, pages 791–796, Pasadena, CA, 2009.
21. A. Felfernig, B. Gula, G. Leitner, M. Maier, R. Melcher, and E. Teppan. Persuasion in Knowledge-based Recommendation. In *3rd Intl. Conf. on Persuasive Technology*, LNCS, pages 71–82. Springer, 2008.
22. A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-based Configuration: From Research to Business Cases*. Elsevier/Morgan Kaufmann Publishers, 1st edition, 2014.
23. A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, and S. Reiterer. Toward the Next Generation of Recommender Systems. In *Multimedia Services in Intelligent Environments: Recommendation Services*, pages 81–98. Springer, 2013.

24. 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.
25. A. Felfernig, W. Maalej, M. Mandl, M. Schubert, and F. Ricci. Recommendation and Decision Technologies For Requirements Engineering. In *ICSE 2010 Workshop on Recommender Systems in Software Engineering (RSSE 2010)*, pages 11–15, Cape Town, South Africa, 2010.
26. A. Felfernig, M. Schubert, and S. Reiterer. Personalized Diagnosis for Over-Constrained Problems. In *23rd International Conference on Artificial Intelligence (IJCAI 2013)*, pages 1990–1996, Peking, China, 2013.
27. 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.
28. H. Garcia-Molina, G. Koutrika, and A. Parameswaran. Information Seeking: Convergence of Search, Recommendations, and Advertising. *Communications of the ACM*, 54(11):121–130, 2011.
29. M. Gasparic and A. Janes. What Recommendation Systems for Software Engineering Recommend: A Systematic Literature Review. *Journal of Systems and Software*, 113:101–113, 2016.
30. J. Golbeck. Computing with Social Trust. Springer, 2009.
31. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
32. P. Grasch, A. Felfernig, and F. Reinfrank. ReComment: Towards Critiquing-based Recommendation with Speech Interaction. In *7th ACM Conference on Recommender Systems*, pages 157–164. ACM, 2013.
33. R. Grünig and R. Kühn. *Successful Decision-Making*. Springer, 2013.
34. A. Gunawardana and G. Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *Journal of Machine Learning Research*, 10:2935–2962, 2009.
35. J. He and W. Chu. A Social Network-Based Recommender System (SNRS). In *Data Mining for Social Network Data*, volume 12 of *Annals of Information Systems*, pages 47–74. Springer, 2010.
36. T. Hennig-Thurau, A. Marchand, and P. Marx. Can Automated Group Recommender Systems Help Consumers Make Better Choices? *Journal of Marketing*, 76(5):89–109, 2012.
37. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Conference on Research and Development in Information Retrieval*, pages 230–237, Berkeley, CA, USA, 1999.
38. C. Huffman and B. Kahn. Variety for Sale: Mass Customization or Mass Confusion? *Journal of Retailing*, 74(4):491–513, 1998.
39. A. Jameson. More than the Sum of its Members: Challenges for Group Recommender Systems. In *International Working Conference on Advanced Visual Interfaces*, pages 48–54, 2004.
40. A. Jameson, B. Berendt, S. Gabrielli, F. Cena, C. Gena, F. Vernero, and K. Reinecke. *Choice Architecture for Human-Computer Interaction*, volume 7. Foundations and Trends in Human-Computer Interaction, 2014.
41. A. Jameson and B. Smyth. Recommendation to Groups. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 596–627. 2007.
42. A. Jameson, M. Willemsen, A. Felfernig, M. de Gemmis, P. Lops, G. Semeraro, and L. Chen. Human Decision Making and Recommender Systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook, 2nd Edition*, pages 611–648. Springer, 2015.
43. D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems – An Introduction*. Cambridge University Press, 2010.
44. Ulrich Junker. QUICKXPLAIN: Preferred Explanations and Relaxations For Over-constrained Problems. In *19th Intl. Conference on Artifical Intelligence*, AAAI'04, pages 167–172. AAAI Press, 2004.
45. D. Kahneman and A. Tversky. Prospect Theory: An Analysis of Decision Under Risk. *Econometrica*, 47(2):263–291, 1979.

46. B. Knijnenburg, N. Reijmer, and M. Willemsen. Each to his own: How Different Users Call for Different Interaction Methods in Recommender Systems. In *RecSys 2011*, pages 141–148, Chicago, IL, USA, 2011.
47. M. Kompan and M. Bielikova. Group Recommendations: Survey and Perspectives. *Computing and Informatics*, 33(2):446–476, 2014.
48. J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.
49. Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, 42(8):30–37, 2009.
50. G. Leitner, A. Fercher, A. Felfernig, K. Isak, S. Polat Erdeniz, A. Akcay, and M. Jeran. Recommending and Configuring Smart Home Installations. In *International Workshop on Configuration 2016 (ConfWS'16)*, pages 17–22, 2016.
51. J. Levin and B. Nalebuff. An Introduction to Vote-Counting Schemes. *Journal of Economic Perspectives*, 9(1):3–26, 1995.
52. G. Linden, B. Smith, and J. York. Amazon.com Recommendations – Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
53. F. Lorenzi and F. Ricci. Case-Based Recommender Systems: A Unifying View. In *International Conference on Intelligent Techniques for Web Personalization*, pages 89–113, 2003.
54. M. Mandl and A. Felfernig. Improving the Performance of Unit Critiquing. In *20th International Conference on User Modeling, Adaptation, and Personalization (UMAP 2012)*, pages 176–187, Montreal, Canada, 2012.
55. M. Mandl, A. Felfernig, E. Teppan, and M. Schubert. Consumer Decision Making in Knowledge-based Recommendation. *Journal of Intelligent Information Systems*, 37(1):1–22, 2010.
56. A. Marchand. *Empfehlungssysteme für Gruppen*. EUL Verlag, 2011.
57. J. Masthoff. Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. *User Modeling and User-Adapted Interaction (UMUAI)*, 14(1):37–85, 2004.
58. J. Masthoff. Group Recommender Systems: Combining Individual Models. *Recommender Systems Handbook*, pages 677–702, 2011.
59. J. McCarthy and T. Anagnost. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts. In *Conference on Computer Support Cooperative Work*, pages 363–372, Seattle, WA, USA, 1998.
60. K. McCarthy, L. McGinty, B. Smyth, and M. Salamó. Social Interaction in the CATS Group Recommender. In *Workshop on the Social Navigation and Community based Adaptation Technologies*, 2006.
61. K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. On the Dynamic Generation of Compound Critiques in Conversational Recommender Systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 176–184. Springer, 2004.
62. K. McCarthy, Y. Salem, and B. Smyth. Experience-Based Critiquing: Reusing Critiquing Experiences to Improve Conversational Recommendation. In *International Conference on Case-Based Reasoning (ICCBR 2010)*, pages 480–494, Alessandria, Italy, 2010.
63. H. Mengash and A. Brodsky. A Group Recommender for Investment in Microgrid Renewable Energy Sources. In *50th Hawaii International Conference on System Sciences*, pages 1485–1494, Hawaii, USA, 2017.
64. G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil. INTELLIREQ: Intelligent Techniques for Software Requirements Engineering. In *Prestigious Applications of Intelligent Systems Conference (PAIS)*, pages 1161–1166, 2014.
65. J. O'Donovan and B. Smyth. Trust in Recommender Systems. In *ACM IUI 2005*, pages 167–174, San Diego, CA, USA, 2005.
66. D. Paraschakis. Recommender Systems from an Industrial and Ethical Perspective. In *10th ACM Conference on Recommender Systems*, pages 463–466, Boston, MA, USA, 2016.
67. J. Payne, J. Bettman, and E. Johnson. The Adaptive Decision Maker. 1993.
68. M. Pazzani and D. Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, 1997.

69. K. Peniwati. Criteria for Evaluating Group Decision-Making Methods. *Mathematical and Computer Modelling*, 46(7–8):935–947, 2007.
70. R. Petty, J. Cacioppo, and D. Schumann. Central and Peripheral Routes to Advertising Effectiveness: the Moderating Role of Involvement. *Journal of Consumer Research*, 10(2):135–146, 1983.
71. L. Recalde. A Social Framework for Set Recommendation in Group Recommender Systems. In *European Conference on Information Retrieval*, pages 735–743. Springer, 2017.
72. R. Reiter. A Theory of Diagnosis from First Principles. *AI Journal*, 32(1):57–95, 1987.
73. F. Ricci and Q. Nguyen. Acquiring and Revising Preferences in a Critique-based Mobile Recommender Systems. *IEEE Intelligent Systems*, 22(3):22–29, 2007.
74. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *10th WWW Conference*, pages 285–295, 2001.
75. G. Sielis, C. Mettouris, G. Papadopoulos, A. Tzanavari, R. Dols, and Q. Siebers. A Context Aware Recommender System for Creativity Support Tools. *Journal of Universal Computer Science*, 17(12):1743–1763, 2011.
76. B. Smith and G. Linden. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017.
77. B. Smyth. Case-Based Recommendation. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *LNCS*, pages 342–376. Springer, Berlin, Heidelberg, 2007.
78. M. Stettinger and A. Felfernig. CHOICLA: Intelligent Decision Support for Groups of Users in Context of Personnel Decisions. In *ACM RecSys'2014 IntRS Workshop*, pages 28–32, Foster City, CA, USA, 2014.
79. M. Stettinger, A. Felfernig, G. Leitner, S. Reiterer, and M. Jeran. Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment. In *20th ACM Conference on Intelligent User Interfaces (IUI2015)*, pages 148–157, Atlanta, Georgia, USA, 2015.
80. N. Tintarev and J. Masthoff. Designing and Evaluating Explanations for Recommender Systems. In *Recommender Systems Handbook*, pages 479–510. 2011.
81. N. Tintarev, J. O'Donovan, and A. Felfernig. Human Interaction with Artificial Advice Givers. *ACM Transactions on Interactive Intelligent Systems*, 6(4):1–10, 2016.
82. A. Valdez, M. Ziefle, K. Verbert, A. Felfernig, and A. Holzinger. Recommender Systems for Health Informatics: State-of-the-Art and Future Perspectives. In *Machine Learning for Health Informatics*, pages 391–414. Springer, 2016.