Dissertation

Group Decision Technologies for Complex Products and Services

Dipl.-Ing. Muesluem Atas, BSc

Graz, June 2020

Institute for Software Technology
Graz University of Technology

Supervisor/First reviewer: Univ.-Prof. Dipl.-Ing. Dr. techn. Alexander Felfernig
Second reviewer: D.Sc. (Tech.) Tomi Männistö

Abstract (English)

Decision technologies such as *Recommender Systems* and *Configuration Systems* support users in identifying a set of useful products and services matching their wishes and needs. Although most existing decision technologies are designed for single users, many scenarios exist in which it is assumed that items are consumed by groups. Examples of these are *visiting a restaurant with colleagues*, *deciding on a travel destination to visit with friends*, or *software release planning with stakeholders in a software project*.

Traditionally, group recommendation approaches are developed based on the well-known recommendation paradigms for individuals, specifically *collaborative filtering, content-based filtering,* and *knowledge-based recommendation.* For instance, in order to generate group recommendations on the basis of a *knowledge-based recommendation* paradigm, the profile of individual group members can be integrated into a group profile. In this strategy, individual group members first articulate their preferences, and then discuss conflicting preferences in order to achieve a consensus within the group. Thereafter, individual preferences are aggregated for generating a group recommendation. The discussion of the articulated preferences of group members plays a vital role, since it helps to increase the probability of discovering the decision-relevant knowledge. After preference articulation and conflict resolution, *aggregation heuristics* are used to merge the preferences of individuals. It is still to some extent unclear, however, which heuristic should be applied in which decision scenario. Besides this issue, (group) recommender systems have suffered from negative influences triggered by decision biases, for example, *GroupThink* or *group polarization effects*. Detecting and counteracting decision biases of this kind has become important as a means of improving the quality of decisions.

As a means of tackling the above listed challenges, we provide the following contributions. We show how constraint-based recommendation as a specific type of knowledge-based recommendation for individuals can be tailored to group scenarios. In addition to this, we introduce an approach for increasing the knowledge exchange among group members for the purpose of discovering the relevant knowledge that will be needed for the making of a high-quality group decision. Moreover, we analyze the prediction quality of different heuristics based on the underlying item domain. Finally, we investigate the influence of decision biases such as *group polarization effects* in different domains (*risk analysis* and *cost estimation*) and propose a solution to counteract such biases.

This work also contributes to the area of *constraint-based recommendation*, which is a type of *knowledge-based recommendation*. *Knowledge-based recommenders* are often applied in complex domains such as

cars, *travel destinations*, and *financial services*. These systems are able to generate recommendations based on knowledge about items, their features, and user requirements. Current research, however, shows that these recommenders face several problems concerning *the identification of a suitable diagnosis (i.e., finding a way to escape the "no recommendation could be found" dilemma) for the whole group when group members' preferences become inconsistent*. Moreover, another challenge is *the identification of a suitable group recommendation from a long list of recommendations*. To bridge these gaps, this thesis introduces approaches which can identify diagnoses and recommendations by considering *fairness among group members*.

Another limitation lies in the knowledge representation of decision problems for complex domains. This thesis introduces an efficient and component-oriented knowledge representation for configuration systems in the *Internet of Things* (IoT) domain. In this context, we show how to apply *Answer Set Programming* techniques to efficiently represent and solve configuration problems.

Abstract (German)

Entscheidungstechnologien zu denen unter anderem *Empfehlungsdienste* und *Konfigurationssysteme* zählen, unterstützen Anwender bei der Identifizierung nützlicher Produkte und Dienstleistungen, die ihren persönlichen Wünschen und Bedürfnissen entsprechen. Obwohl die meisten bestehenden Entscheidungstechnologien für Einzelbenutzer konzipiert wurden, gibt es viele Szenarien, in denen Entscheidungen von Gruppen getroffen werden. Beispiele hierfür sind *der Besuch eines Restaurants mit Kollegen und Kolleginnen* oder *die Planung von Software-Releases mit Stakeholdern in einem Softwareprojekt*.

Gruppenempfehlungsansätze basieren traditionell auf den bekannten Empfehlungsparadigmen für Einzelpersonen, wie bspw. *Collaborative Filtering, Content-based Filtering* und *Knowledge-based Recommendation*. Um Gruppenempfehlungen bspw. auf der Grundlage von *Knowledge-based Recommendation* zu generieren, kann das Profil einzelner Gruppenmitglieder zu einem Gruppenprofil aggregiert werden. Diese Strategie beginnt mit der Artikulation der Präferenzen aller Mitglieder und geht anschließend in eine gemeinsame Diskussion über. In dieser Diskussion werden dann widersprüchliche Präferenzen besprochen, um einen gemeinsamen Konsens zu finden. Anschließend werden die individuellen Präferenzen der Mitglieder mit Hilfe von *Heuristiken* aggregiert, um daraus eine Gruppenempfehlung zu generieren. Wesentlich für eine gelungene Gruppenentscheidung ist die Diskussion zwischen den verschiedenen Mitgliedern. Es ist jedoch noch unklar, welche *Heuristik* in welchem Entscheidungsszenario angewendet werden soll. Außerdem leiden Empfehlungssysteme unter negativen Einflüssen, die durch *Entscheidungsverzerrungen* ausgelöst werden. Das Erkennen solcher Verzerrungen und Gegensteuern sind wichtig, um die Qualität von Entscheidungen zu verbessern.

Um die oben genannten Probleme zu lösen, werden folgende Beiträge im Rahmen dieser Arbeit vorgestellt. Zu Beginn stellen wir Ansätze zur Minimierung von *Entscheidungsverzerrungen* vor. Darüber hinaus präsentieren wir einen weiteren Ansatz, um den Wissensaustausch zwischen Gruppenmitgliedern zu verbessern, um optimale Entscheidungen zu ermöglichen. Des Weiteren analysieren wir die Vorhersagequalität von verschiedenen *Heuristiken* basierend auf der zugrunde liegenden Objektdomäne. Schließlich untersuchen wir den Einfluss von Entscheidungsverzerrungen wie *Gruppenpolarisierungseffekte* in zwei Bereichen (*Risikoanalyse* und *Kostenschätzung*) und stellen eine Lösung vor, um solchen Verzerrungen entgegenzuwirken.

Diese Arbeit leistet auch einen Beitrag zum Thema *Constraint-basierte Empfehlungen*. Constraint-basierte Empfehlungen zählen zu den *Knowledge-based Empfehlungsmethoden* und werden häufig in komplexen

Bereichen (z.B. *Reiseziele*) angewendet. Die generierten Empfehlungen basieren auf Informationen über Benutzeranforderungen, Artikel und deren Eigenschaften. Aktuelle Forschungsergebnisse zeigen allerdings, dass solche Empfehlungssysteme mit mehreren Problemen konfrontiert sind: z.B. *wie geht man mit dem "keine Empfehlung gefunden" Dilemma um*? D.h. wie findet man eine geeignete Diagnose für eine Gruppe, wenn die Präferenzen der Mitglieder sich widersprechen. Eine weitere Herausforderung ist die Identifizierung einer geeigneten Empfehlung aus einer großen Anzahl von Alternativen. Um diese Lücken zu schließen, stellen wir geeignete Ansätze vor mit denen Diagnosen und Empfehlungen ermittelt werden können, welche *Fairness* unter Gruppenmitgliedern garantiert.

Eine weitere Einschränkung von Entscheidungstechnologien ist die ineffiziente Repräsentation von Entscheidungsproblemen. Um dies zu vermeiden, stellen wir in unserem letzten Beitrag eine effiziente und komponentenorientierte Wissensrepräsentation für Konfigurationssysteme im Bereich des *Internet of Things* vor. In diesem Zusammenhang zeigen wir, wie man *Answer Set Programming* anwendet, um Konfigurationsprobleme intuitiv darstellen und lösen zu können.

Acknowledgement

First of all, I would like to thank my supervisor, Univ.-Prof. Dipl.-Ing. Dr.techn. Alexander Felfernig who offered me this position, has been motivated me during my study and enriched me with his broad knowledge.

Additionally, I am very grateful to my colleagues M.Sc.Thi Ngoc Trang Tran, Dipl.-Ing. Dr.techn. Martin Stettinger, Dipl.-Ing. Ralph Samer, M.Sc. Seda Polat-Erdeniz, Stefan Sgouridis, Michael Jeran, Dipl.-Ing. Dr.techn. Stefan Reiterer, Dipl.-Ing. Thorsten Ruprechter, and Dipl.-Ing. Christoph Uran who have contributed to my study.

Lastly, I want to deeply thank my family and my best friends Melanie Merve Seker, Dipl.-Ing. Firat Kilic, and Dipl.-Ing. Labinot Xhafa who have never stopped to support me during the last years.

Muesluem Atas
Graz, 2019

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Graz, _____ _____

Place, Date Signature

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Graz, am _____ _____

Ort, Datum Unterschrift

Contents

Chapter 1

Introduction

1.1. Background and Motivation

Users frequently tend to opt for a range of different strategies when deciding what to consume, due to the difficulties they encounter in identifying useful items or services. Decision-making strategies traditionally *use information from different sources,* such as *friends, colleagues, popular items chosen by the crowd*, or *the advice of an expert*. Empirical observations show, however, that the application of these strategies does not always guarantee affordable, personal, and high-quality recommendations. Therefore, for the sake of optimal recommendations which meet the user requirements, the application of intelligent techniques is essential (Jannach et al., 2010). In brief, the identification of an optimal decision appears to be an altogether challenging and time-consuming task, since users must first analyze a large set of available products or services, then apply different decision strategies, and finally try to identify the most suitable item. Identifying optimal recommendations efficiently has thus become a matter of some urgency in recent years (Jannach et al., 2010).

In order to automate these decision strategies and to provide high-quality recommendations efficiently, "recommender systems" have been applied in many fields (Burke, 2000; Jannach et al., 2010; Burke et al., 2011; Ricci et al., 2011). These systems support users in identifying products and services that meet their desires and needs. Recommender systems are applied today in several domains such as tourism (Loh et al., 2003; Borràs et al., 2014), financial products (Felfernig et al., 2007; Dávid, 2016), real estate (Yuan et al., 2013), movies (Christakou et al., 2007; Azaria et al., 2013), news (Liu et al., 2010), web sites (Wang et al., 2008), and in many other domains. The goal of these systems is to understand the human logic and decision psychology of users (i.e., consumers), which is the premise to generate recommendations satisfying users' preferences. On the technical level, several different concepts are already available for exploring the information and knowledge that are necessary for making a good decision (Jannach et al., 2010). On the psychological level, these systems must be capable of understanding the decision psychology of users. Without an understanding of human psychology, a recommender system cannot generate accurate explanations as to *why and how a specific item is recommended to the active user*. In the event of a misunderstanding/misinterpretation of human psychology, recommender systems are thus unable to generate personal and high-quality recommendations and as a result the trust of users in such systems decreases. In order to understand human decision making (Ricci et al., 2010), the following key

questions must be answered: *"Which factors influence the decision of a user?"*, *"How are decisions made?"*, and *"How do decisions get biased by other sources?"* It would appear that making decisions to satisfy user preferences has remained an extremely challenging task (Ricci et al., 2010).

Recommender systems can be categorized into *collaborative filtering, content-based filtering, knowledge-based recommendation, hybrid recommendation*, and *group recommendation* (Goldberg et al., 1992; Burke, 2000; Meteren and Someren, 2000; Burke, 2002; Jannach et al., 2010; Masthoff, 2011; Felfernig et al., 2017a, 2018d) (for more details, see Chapter 2).

Collaborative filtering recommends items to individual users on the basis of users with similar tastes (Goldberg et al., 1992; Konstan et al., 1997). Some well-known companies such as Amazon*, Netflix†, and Apple‡ have used this technology for many years to recommend items to their customers. Collaborative filtering is based on the following idea: *"If users shared the same preferences and tastes in the past, they will also have similar preferences and tastes in the future."* Usually, this approach does not require any information or knowledge about the products or services themselves. Only the information about users and their consumed items is needed for making recommendations. This approach is implemented on the basis of the following two steps: First, the similarities between user behavioral patterns are defined (i.e., similarity of users based on their consumed items), and then these behavioral patterns are used to make predictions for the active user.

Content-based filtering compares the active user's profile with the description of available items (Pazzani and Billsus, 1997; Mooney and Roy, 2000; Jannach et al., 2010). The profile of the active user and the description of items are usually represented as tags or keywords/categories. A learning algorithm then generates the active user's profile based on his/her consumed items, and finally personal recommendations are identified based on the active user's profile. In contrast to *collaborative filtering*, this approach does not require large user groups to generate high-quality recommendations, since it compares only the active user's profile with the description of new items.

Knowledge-based approaches can be effectively used in domains where *collaborative-* and *content-based filtering* cannot be applied. Knowledge-based recommenders are especially helpful for complex domains in which items are not consumed very often. Examples of such domains are *computers, financial services, real estate, travel destinations*, and *cars* (Torrens et al., 2003; Peischl et al., 2009; Leitner et al., 2012; Reiterer, 2015; Reiterer et al., 2015; Win and Srisura, 2019). Knowledge-based recommenders follow the idea of making recommendations based on the knowledge about items, their features (i.e., knowledge base), and user requirements (i.e., user constraints). There are two different types of knowledge-based recommender systems: *critiquing-based* which are also considered as specific type of *case-based* recommenders (Burke, 2000; Bridge et al., 2005) and *constraint-based recommender systems* (Felfernig and Burke, 2008; Jannach et al., 2010; Felfernig et al., 2011a). Both types of recommenders are very similar and differ only in the way in which they determine recommendations (i.e., solution). Case-based recommender systems in particular determine solutions by using *similarity metrics* whereas constraint-based recommender systems apply constraint search or conjunctive query-based approaches. In such recommenders, the recommended item must fulfill all the criteria defined by the user and the knowledge base. If user constraints are

*Amazon `https://www.amazon.com/`
†Netflix `https://www.netflix.com/`
‡iTunes `https://www.apple.com/itunes/`

consistent with the underlying constraint set (i.e., knowledge base), several recommendations can be generated. However, it is still unclear how to effectively identify the most suitable solution from a long list of recommendations (Atas et al., 2018c).

If user preferences are inconsistent with the underlying constraint set, users will need to be supported in finding a way to escape the *"no recommendation could be found"* dilemma (Reiterer et al., 2015). In such cases, some of the preferences have to be deleted or adapted such that a recommendation can be determined (i.e., a diagnosis has to be identified). In this context, an adequate diagnosis based on the constraints articulated by a user must be identified. In group recommendation scenarios, diagnosis finding is even more challenging since a *suitable* recommendation for the group has to be identified. If the preferences of group members are inconsistent with the underlying constraint set, a diagnosis should be identified, which takes into account the preferences of all group members (as far as this is possible). If the recommended solution only considers the preferences of some of the group members, group satisfaction will decrease which can negatively influence the mood of the group (Lind et al., 2001). Therefore, an optimal group recommendation can only be acquired if the fairness within a group and the satisfaction of all group members are sufficiently taken into account (Atas et al., 2019a,b).

As briefly mentioned above, each recommendation approach (*collaborative filtering, content-based filtering,* and *knowledge-based recommendation*) has its strengths and weaknesses. In order to exploit synergy effects, *hybrid recommendation approaches* have been proposed (Burke, 2002; Jannach et al., 2010; Ricci et al., 2010). The idea of these approaches is to achieve high-quality recommendations on the basis of combining basic recommendation approaches. For example, collaborative filtering shows a cold-start problem when new products or services are available, whereas content-based filtering can overcome this problem because it recommends items based on their description.

Although different variants of recommender systems have been proposed for individuals, many scenarios exist in which *items are consumed by a group of users* (Masthoff, 2011; Felfernig et al., 2018d). Some examples of this are choosing a *restaurant* to have dinner at with colleagues, selecting a *movie* to watch with the family, deciding on a *travel destination* to visit with friends, or *software release planning* with stakeholders in a software project. One of the key aspects of a good group decision is the knowledge exchange among group members, because the intensive analysis of all alternatives (i.e., the discussion of all alternatives) will disclose *hidden profiles* and increase the decision quality (Stasser and Titus, 1985). The higher the frequency of the exchanged information among group members, the better the quality of the corresponding decision (Wittenbaum et al., 2004). Moreover, through the increased frequency of information exchange among group members, the probability of discovering the relevant knowledge (the knowledge of some group members, which is not known by the other group members) also increases, which is essential for a high-quality group decision (Wittenbaum et al., 2004). Group diversity (such as demographic and educational background) or recommendation diversity are, for example, possibilities to increase the knowledge exchange among group members. However, it is still not completely clear how recommendation diversity influences the information exchange among group members (Atas et al., 2017).

After exchanging the relevant knowledge among group members, a strategy (*aggregated predictions* or *aggregated models*) is used to aggregate preferences of individual group members (for more details, see Section 2). Different approaches exist for the determination of group recommendations (Jameson and Smyth, 2007) in which the preferences of individual group members are aggregated based on aggregation

functions (also termed *aggregation heuristics* or *aggregation strategies*). There are a number of different aggregation heuristics available (e.g., *most pleasure, least misery,* and *average voting*) that are applied in group recommendation scenarios (Masthoff, 2011). It is still unclear to some extent, however, which heuristics should be applied and in what context, or to what extent the item domain influences how appropriate the group aggregation heuristics are (Felfernig et al., 2017a).

Apart from open issues in preference aggregation, group recommender systems have also suffered from the negative influences of decisions biases (Jameson, 2004; Jameson and Smyth, 2007; Berkovsky and Freyne, 2010; Masthoff, 2011; Felfernig et al., 2017a, 2018d). For instance, compared to individual recommendations, recommender systems designed for groups must tackle additional decision biases such as *anchoring effects, GroupThink*, or *group polarization effects* (Janis, 1972; Myers and Lamm, 1976; Stettinger et al., 2015a).

Anchoring effects (Jacowith and Kahneman, 1995; Adomavicius et al., 2011; Stettinger et al., 2015a) occur during the decision making process where user decisions can be influenced by an initially available information. This information is often denoted as the *"anchor"*. In group decision scenarios, the preference of the first articulating group member can influence the preferences of the others. In order to counteract anchoring effects in such scenarios, the preferences of group members should not be disclosed in the early stages of the group decision making process (Stettinger et al., 2015a).

GroupThink is a psychological phenomenon which was initially analyzed by Irving Janis (Janis, 1972). In group decisions, group members often have diverse opinions concerning the given alternatives (i.e., items). However, group members often do not articulate their preferences to avoid conflicts or reaching a consensus within the group. The desire for group conformity leads instead to a suboptimal decision outcome, because "*individual creativity*" and "*independent thinking*" are lost. This phenomenon, which is termed *GroupThink*, deteriorates nearly every group decision (Esser, 1998). For example, meetings in companies typically involve the whole team, but decisions are only made by some team members (leaders, project managers, etc.). In order to avoid the GroupThink phenomenon, leaders should not immediately articulate their opinions and analyze alternative solutions of other team members in detail. Moreover, the opinion of experts outside the group can be integrated to achieve a high-quality decision.

Group polarization is the tendency of a group to make decisions that are more extreme than the average of the individual group members' preferences (Stoner, 1961; Myers and Lamm, 1976; Sunstein, 2002; Adomavicius et al., 2011). This phenomenon indicates that if individual group members tend to make risky decisions, then the group decision will be even riskier. Also, if they tend to make cautious decisions, then the group decision will be more cautious. This phenomenon was discovered by James Stoner in 1961 when he tried to compare the risk-taking of individuals and groups (Stoner, 1961). These effects are usually triggered by the "GroupThink" bias in which some of the group members don't want to articulate their own opinions that other group members may disagree with (Whyte, 1989). This bias can influence decisions in many fields such as *politics, sports, financial services,* and *business* (Sunstein, 2002). Some of the studies made in this area have investigated the influence of polarization effects on a specific domain (e.g., *risk analysis*) and the insight into these effects has been not discovered in some other domains (e.g., *cost estimation*). Moreover, no results have been published on how to avoid these effects on the group decision quality (Atas et al., 2018a).

Apart from the open issues in decision biases, group members might also face situations in which they must make a decision in complex domains such as *Requirements Engineering*. In this domain, the application of recommendation techniques is not sufficient to support users in making decisions on all *Requirements Engineering* activities. The core activities of *Requirements Engineering* are *the definition and elicitation of requirements, the identification of stakeholders, the negotiation of requirements, the identification of dependencies,* and *release planning*. *Requirements Engineering* plays a crucial role in the success of software projects, especially for scenarios in which the size and complexity of software projects is rapidly increasing. In this context there is great demand for applying automated and intelligent techniques in each Requirements Engineering core activity (Hofmann and Lehner, 2001; Mobasher and Cleland-Huang, 2011; Felfernig et al., 2017c). For instance, recommender systems can be applied in the following core activities of *Requirements Engineering*:

- Definition and elicitation of requirements: In this activity, group members (also referred to as *stakeholders*) must define software requirements. In this context, a recommender system can identify reusable requirements that have already been defined in previous projects and may be relevant to the current project (Dumitru et al., 2011).
- Stakeholder identification: The main focus of this activity is to identify a group of persons who are capable of providing a complete and accurate description of software requirements. In this context, a recommender system can support stakeholders' identification on the basis of social network analysis (Lim et al., 2010). For instance, based on the description of requirements, a *content-based recommender system* can identify stakeholders who completed similar tasks in previous projects.
- Negotiation of requirements: In this step, a group of stakeholders must jointly evaluate the quality of requirements and also try to figure out in which way the requirements should be taken into account in release planning. Group recommendation techniques can be applied in this activity to assist stakeholders with the requirement evaluation and prioritization process (Ninaus et al., 2014a).
- Identification of dependencies: *Dependency identification* is a crucial activity in Requirements Engineering. During this phase, stakeholders must find and define correct dependencies as early as possible (Li et al., 2012). In the case of incomplete or incorrect dependencies, additional efforts are triggered in terms of redesign and reprogramming. An example for a dependency is "R_1 *requires* R_2" ($R_1 \rightarrow R_2$) which indicates that for the realization of requirement R_1, requirement R_2 must first be implemented. Many projects today involve a large number of requirements and the identification of dependencies can be challenging for stakeholders. Due to this fact, there is an urgent need for automated technologies to assist stakeholders in finding dependencies between requirements (Atas et al., 2018b).

Another core activity of Requirements Engineering is *Release Planning*. After having completed the requirements elicitation, all the requirements need to be evaluated according to different requirement properties and assigned to different releases by the stakeholders. The most important requirements should be part of the first release which can be delivered to end-users as a "*minimal viable product*". By contrast, requirements which are evaluated as less important will be developed in later releases. Unfortunately, recommender systems do not support scheduling activities such as release planning. In this context, *configuration systems* were developed to support users in configuring complex products or services, which enable individual customization of products (Stumptner, 1997; Aldanondo and Vareilles, 2008; Felfernig et al., 2014a,b).

Furthermore, configuration systems are applied to complex domains, such as *cars* (Macher et al., 2015), *personal computers* (Felfernig et al., 2014a), *financial services* (Stolze et al., 2000; Felfernig et al., 2007), and in many other domains. These systems allow individual customization of complex products and services (i.e., designed for highly variant consumer items). Configuration technologies are becoming increasingly popular today in many different *Internet of Things (IoT)* scenarios (Atzori et al., 2010). In the IoT context, configurators can be applied in many scenarios. For example, for the identification of ramp-up configurations, these systems need to figure out the components that will be needed in a specific IoT setting. The knowledge base for configurations of this type, however, are often represented as a constraint satisfaction problem, which is neither intuitive nor easy to maintain for complex decision problems. As a result, there is a need for efficient knowledge representation concepts (Felfernig et al., 2017b).

1.2. Research Objectives

We discussed the challenges of recommender and configuration systems in the previous section. On the basis of these challenges, our research objectives are introduced in the following:

1. **Group recommendations on the basis of recommendation paradigms for individuals**
   Recommender systems are effective tools to support users in identifying a set of useful items matching their wishes and needs. Many recommendation technologies and concepts are now available, since these systems have already been applied for a couple of years (Resnick and Varian, 1997). Although most existing systems are designed for recommending products and services to single users, there are many scenarios in which recommendations need to be determined for groups. Examples of these are *deciding on a restaurant to have dinner at with your friends*, *deciding on a digital camera to use together with your partner*, or *deciding on release planning with stakeholders in a software project*. Unfortunately, until now, some recommendation concepts and technologies such as *constraint-based recommenders* have not been well-designed for groups. In order to facilitate the development of such recommenders, group recommendation strategies have to be implemented. The application of recommendation concepts and technologies in group scenarios, however, triggers additional challenges. For instance, in some decision scenarios (e.g., deciding on a restaurant), social aspects such as *fairness* and *consensus among group members* should be taken into account (i.e., the preferences of all members should be taken into account as far as possible when generating a group recommendation). The open issues mentioned above raise the first research question:

   ($Q1$) *How can constraint-based recommendation for individual users be implemented for group scenarios?*

2. **Group recommendation strategy on the basis of item domain**
   As already stated, group recommender systems generate recommendations by aggregating the preferences of all individual group members. There are two basic strategies for this: *aggregated predictions* and *aggregated models*. In the first of these, the recommender system suggests recommendations (ratings or items) for individual group members and these recommendations are then aggregated in order to generate group recommendations. The application of such strategies, however, cannot guarantee the satisfaction of all group members, especially in situations where the recommendations generated for individuals are diverse (i.e., do not overlap). In such cases, the recommendation only

satisfies some group members and this decreases the overall satisfaction of the group. The follow-up discussion of the group members thus takes on an important role in the decision making process, since no ranking of the individual candidate items is provided. Another type of *aggregated prediction* strategy is to aggregate group-member-specific predictions for candidate items (e.g., aggregation of rankings). The outcome of this approach is a ranking of candidate items.

In the second strategy, the profiles of individual group members are aggregated into a group profile, and the recommendations are then generated based on the group profile. This approach is usually applied in scenarios where all the group members attempt to define a group profile based on the negotiation and adaptation process for their preferences that has the target of a consistent group profile. As with the first strategy, this strategy also has some disadvantages, for instance, biases such as *GroupThink* and *Anchoring Effects* are triggered by the disclosing of individual preferences in the early stages of the group decision making process (Janis, 1972; Felfernig, 2014). After deciding on a strategy, *group aggregation heuristics* (also often referred to as *group aggregation functions* or *group aggregation strategies*) are used to merge individual group member profiles or the individual recommendations of group members. On an abstract level, these aggregation functions can be grouped in three different categories: *majority-based* (e.g., Approval Voting, Copeland Rule), *consensus-based* (e.g., Average Voting, Fairness), and *borderline* (e.g., Least Misery, Most Pleasure). Although a few studies have been made on group aggregation heuristics, it is still to some extent unclear which aggregation heuristic can be used for what scenario. In other words, *"which aggregation heuristic or recommendation strategy can be applied on the basis of item domain"* is still an open issue. This gap brings us to the second research question:

($Q2$) *Which recommendation strategy should be applied in which item domain?*

3. **Recommendations to stimulate information exchange in group decisions**
   Group decisions are often made after a detailed discussion of alternatives among group members. As already mentioned in Section 1.1, the more information that is exchanged among group members, the higher the quality of the decision will be. Group decisions are frequently not optimal, because information relevant for making them that is, possessed by individual group members, is not shared with the others (i.e., *hidden profiles*) (Stasser and Titus, 1985; Greitemeyer and Schulz-Hardt, 2003). Hidden profiles are usually shared by some of the group members and each member individually has only a subset of information. Hence, the degree of knowledge exchange within a group can have a major impact on the decision quality. In this context, intelligent techniques are needed that help to increase the knowledge exchange among group members. Before answering the question of *"how to increase the knowledge exchange among group members?"*, however, we must first answer the following question: *"What are the factors that trigger a group discussion and increase knowledge-exchange among group members?"*. One alternative is to form a diverse group or to provide diverse recommendations (Yaniv, 2011). For instance, if suggestions to a group are too diverse, this diversity can trigger additional group discussions about decision alternatives. These open issues lead to the third research question:

   ($Q3$) *How does recommendation diversity influence the information exchange among group members?*

4. **Polarization effects in group decisions**
   As discussed above, group decisions are affected by some decision biases in a manner similar to single user decisions. Some of these decision biases such as *anchoring effects and GroupThink* are well-known and have also been well analyzed in different domains (Janis, 1972; Stettinger et al., 2015a). This said, there are, however, many other group decision biases such as *group polarization effects*, which have not been adequately analyzed (Roy and Yan, 2009; Li and Luo, 2011). As stated above, group decisions are often made after a detailed discussion of the decision alternatives. Indeed, group members have different opinions regarding decision alternatives, and a discussion might therefore be initiated in order to let group members express their opinions. However, *GroupThink* can cause a situation in which some group members do not want to articulate their opinions. This bias can minimize conflicts among group members and accelerate the consensus making process. At the same time, it can lead to group decisions that are more extreme than the initial opinions of its members (Stoner, 1961). An example of this can be observed in the *politics* domain when the members of a political party make decisions (e.g., right-wing or left-wing political parties). Since all members of a political party share the same (or similar) opinions regarding a decision, the group decision becomes more extreme than the initial opinions of the individual group members. Existing studies have investigated the influence of polarization effects in some specific domains (e.g., *risk analysis*) and the resulting insights could not be discovered in other domains (e.g., *cost estimation*). Furthermore, approaches to counteract polarization effects do not exist. These open issues motivated the fourth research question:

   (*Q*4) *How do "Group Polarization Effects" influence the outcome of group decisions and how can these effects be counteracted?*

5. **Socially-aware recommendation for over-constrained problems**
   Knowledge-based recommender systems are recognized as helpful tools for recommending complex items such as *cars, real estate,* or *travel destinations* to users (Torrens et al., 2003; Reiterer, 2015). There are two different types of these systems: *case-based* and *constraint-based* recommender systems (Bridge et al., 2005; Felfernig and Burke, 2008). Constraint-based recommender systems can apply constraint solving to generate recommendations. In this approach, the recommended item must fulfill all the constraints defined by the active user and the recommendation knowledge base. For cases in which items are to be recommended for a group of users in particular, the recommendation process becomes more complex because of the conflicting preferences of the users. In a group scenario, there can be conflicts between the group members' preferences (i.e., individual constraints) or between the preferences of the group members and the recommendation knowledge base. If the preferences of the group members are inconsistent with the underlying knowledge base, then support is needed for finding a way out of the *"no recommendation could be found"* dilemma (i.e., identifying a diagnosis) (Felfernig et al., 2009a). A diagnosis suggests a solution to adapt or remove some of the constraints, which helps to find at least one recommendation. The identification of a *suitable* diagnosis for the group is a cumbersome process because the identified diagnosis must be fair to all group members and should not require only the adaption or removal of the constraints of some group members. Fairness among the group members together with satisfaction for them all in a commonly consumed item is vital for an optimal group decision. Recommending items to a group with constraint-based recommender systems appears to be a complex process. In order to streamline this process, *group aggregation functions* (also called *group aggregation heuristics*) can

be applied to recommend items for groups in scenarios where the constraints of group members become inconsistent (i.e., over-constrained). In our approach, we first calculate the similarities between group members' preferences and the available items, and then apply a *suitable* aggregation function to these similarities in order to recommend an item to the group. It is still to some extent unclear, however, which aggregation function is the most suitable for predicting items to a group. This open issue raises the following research question:

($Q5$) *Which aggregation functions are suitable for predicting items to groups in situations where the preferences of group members become inconsistent?*

6. **Socially-aware diagnosis for constraint-based recommendation**
   As described in the previous research question, inconsistencies between group member preferences in constraint-based recommendation scenarios have to be resolved by taking into account the preferences of all the group members. In order to tackle this challenge, a diagnosis must be identified, which does not decrease the group satisfaction, while simultaneously taking *fairness* among the group members into consideration. In the previous research question, the challenge lay in the generation of a group recommendation without the identification of diagnoses by using group aggregation functions. Scenarios also exist in which the diagnosis identification is desired (i.e., first, a group of users wishes to analyze all possible diagnoses, and then selects a *suitable* diagnosis for the group). A diagnosis is a minimal set of group member preferences, which must be adapted or deleted in order to find a recommendation (Reiter, 1987). Traditionally, diagnoses are determined based on *Breadth-First Search* and focus on the identification of a diagnosis of minimal cardinality (Reiter, 1987; Felfernig et al., 2004). However, a diagnosis with minimal cardinality is not necessarily the most suitable diagnosis for a group. For example, should a diagnosis with minimal cardinality include only the preferences of a single group member, this group member alone must adapt or delete his/her preferences, whereas the preferences of other group members can remain unchanged (i.e., unfair strategy; the satisfaction of all group members is not guaranteed). To counteract this problem, socially-aware diagnosis techniques, which help to identify diagnoses that best match the preferences of all group members, are required. This leads to the sixth research question:

   ($Q6$) *How to identify a socially-aware diagnosis when group members' preferences are inconsistent?*

7. **Similarity-aware recommendations for constraint-based recommender systems**
   Constraint-based recommender systems allow the individual customization of complex industrial products and services in order to satisfy individual user needs. In some cases, user preferences can be inconsistent with the underlying constraint set and users need to be supported in finding a diagnosis. In addition, after identifying and applying a diagnosis, the proposed recommendation should meet the defined user requirements in order to increase the user satisfaction. Furthermore, recommender systems in complex-product domains (e.g., *personal computers* or *cars*) are usually capable of calculating millions of suggestions. It is also often the case that these systems must deal with difficult problems such as *system maintainability, consistency maintenance,* and *efficient response times*. One of the major challenges faced by these recommenders is to identify the most suitable recommendation from a long list of candidates to increase user satisfaction. This means the identified recommendation should be similar to the constraints articulated by a user and must be computed with an acceptable performance. A simple solution is the comparison of user constraints with each

possible recommendation in order to identify the most similar recommendation. Due to unacceptable runtimes, such naive solutions are not applicable. The related research question is the following:

(*Q*7) *How can a constraint-based recommender system identify a recommendation which is similar to the preferences articulated by a user?*

8. **Automated identification of dependencies between requirements**
   The size and complexity of software projects has increased rapidly in the past decade. As a result, an increasingly strong demand has emerged for applying automated and intelligent techniques to support core activities for Requirements Engineering processes. The core activities of a *Requirements Engineering* process are *the definition and elicitation of requirements, the negotiation of requirements, the identification of dependencies, the identification of stakeholders, and release planning* (Hofmann and Lehner, 2001). As already stated in Section 1.1, *recommender systems* and *configuration systems* can be applied in many core activities of "Requirements Engineering" in order to achieve high-quality decisions. For instance, *dependency identification* is a crucial activity in Requirements Engineering and the defined dependencies between requirements have to be accurate, complete, and consistent. Many projects today are characterized by a large number of requirements. As a result, the identification of dependencies among the requirements becomes a major challenge for the people involved. Due to this limitation, there is an urgent need for automated techniques, which can assist stakeholders in finding the dependencies among the requirements. These open issues raise the following research question:

   (*Q*8) *How can requirement dependencies be identified automatically using supervised classification techniques?*

9. **Knowledge representations for IoT configuration scenarios**
   As has been pointed out, *configuration systems* support users in scenarios where the individual customization of products or services is needed. These systems configure an item based on a set of predefined component types and corresponding constraints that restrict the way in which component instances can be combined. In addition, requirements (i.e., constraints) articulated by users restrict the set of possible solutions. Configuration technologies are popular and applied in many domains such as the *Internet of Things (IoT), operating systems,* and *software product lines* (Stumptner, 1997). In the context of IoT, configurators can be applied to many scenarios, for example, in the identification of ramp-up configurations, which is the process of figuring out the components that will be needed for a specific IoT setting. Such IoT settings are complex and usually involve hundreds of constraints and components. Such complex configuration scenarios, however, are mostly implemented on the basis of constraint-based approaches (Hotz et al., 2014; Felfernig et al., 2014a). Unfortunately, knowledge representations of this kind are relatively hard to maintain and this gap triggers an urgent need for alternative knowledge representations. This leads to the following research question:

   (*Q*9) *How can configuration knowledge be efficiently represented in the IoT domain?*

1.3. Contributions

The relevant research questions of this thesis were presented in the previous section (see Section 1.2). In this section, we present the main contributions of this thesis which are briefly introduced in Table 1.1.

Research questions	**Contributions**
($Q1$) *How can constraint-based recommendation for individual users be implemented for group scenarios?*	In this context, basic group recommendation techniques are introduced (Felfernig et al., 2018a). We present real-world scenarios for group recommendation and show how basic recommendation concepts such as *collaborative filtering, content-based filtering, constraint-based recommendation, critiquing-based*, and *hybrid recommendation* can be applied to support group recommendation. In addition, we classify these recommendation concepts into the following aggregation strategies: **Aggregated predictions:** There are two basic approaches to aggregate predictions. (1) Recommendations for individual group members are generated and then merged together for the recommendation of items to groups. (2) Group-member-specific predictions for candidate items are aggregated. The outcome of this approach is a *ranking of candidate items*. **Aggregated models:** First, individual group member profiles are merged together to generate a group profile. Then, recommendations on the basis of the group profile are generated for the group. In this context, our contribution is to provide an overview of group recommendation techniques and algorithms to researchers in the field of recommender systems. In particular, we show how *constraint-based recommenders* can be designed for groups of users.

(*Q*2) *Which recommendation strategy should be applied in which item domain?*	We analyze the prediction quality of group recommendation strategies (*Most Pleasure, Least Misery, Average Voting, Minimal Group Distance, Ensemble Voting, Multiplicative*) depending on the item type. We separate item types based on their decision effort as follows: *high-involvement items* (i.e., items with high related decision effort such as *deciding on an apartment to share with friends for many years*) and *low-involvement items* (i.e., items with low related decision effort such as *deciding on a restaurant to have dinner at with your friends*). Moreover, we conduct a user study and collect a dataset from 420 participants in order to analyze the appropriateness of various aggregation strategies. For the two item domains of restaurants and shared apartments, we show that aggregation strategies in group recommendations should be tailored based on the underlying item domain (i.e., different aggregation strategies have to be applied based on the item domain) (Felfernig et al., 2017a).
(*Q*3) *How does recommendation diversity influence the information exchange among group members?*	Due to the fact that knowledge exchange among group members discloses *hidden profiles* (Stasser and Titus, 1985; Greitemeyer and Schulz-Hardt, 2003) and increases the decision quality, we investigate the possibilities of exploiting recommendation technologies to foster intended behavior, which can also be interpreted as a kind of persuasive technology. These possibilities are used to form diverse groups (e.g., group members with different educational backgrounds) or to provide diverse recommendations. We analyze different recommendation strategies with a varying degree of recommendation diversity and investigate the impact of recommendation diversity on the knowledge interchange between users. The outcome shows that recommendation diversity has an impact on the frequency of information exchange between group members. The higher the diversity of recommendations, the more information is exchanged among group members (Atas et al., 2017).

(*Q*4) *How do "Group Polarization Effects" influence the outcome of group decisions and how can these effects be counteracted?*	In the context of group recommendations, we analyze the existence of *Group Polarization Effects* in the following two dimensions: "*Risk*" and "*Cost estimation*". In the *risk* dimension, we show that if individual group members tend to make cautious decisions, then the group decision will be more cautious (i.e., *Group Polarization Effects* exist). However, if individual group members tend to make risky decisions, then the group decision is not riskier (i.e., *Group Polarization Effects* do not exist). In the *cost estimation* dimension, we find out that group polarization effects only exist at the lower boundaries of the cost range (i.e., if individual group members tend to estimate lower costs, then the cost estimation of the group is much lower). Furthermore, we present a way to counteract the group polarization bias in order to prevent the deterioration of decision quality. To counteract group polarization effects, counterarguments regarding a decision can be provided to prevent an extreme group decision outcome. These arguments can be automatically generated by a system or by a new group member who has a different opinion from others (Atas et al., 2018a).
(*Q*5) *Which aggregation functions are suitable for predicting items to groups in situations where the preferences of group members become inconsistent?*	We present a constraint-based socially-aware recommender system and analyze the prediction quality of some preference aggregation functions applied to generate group recommendations, such as *consensus-based* (Average Voting, Minimal Group Distance, Multiplicative, and Ensemble Voting) and *borderline* (Least Misery and Most Pleasure). Furthermore, we analyze their capability to predict relevant items in situations where no solutions could be found for a given set of preferences (i.e., over-constrained items). The result indicates that consensus-based aggregation functions which consider all group members' preferences lead to a higher prediction quality compared to borderline aggregation functions, which focus solely on the preferences of some individual group members (Atas et al., 2018c).

(*Q*6) *How to identify a socially-aware diagnosis when group members' preferences are inconsistent?*	Knowledge-based group recommender systems are usually used for recommending complex items (i.e., items with a high related decision effort which is also termed *high-involvement items*) to a group of users. These systems can generate recommendations only if constraints given by group members and knowledge base are consistent. Otherwise, in case of an inconsistency, the identification of a suitable diagnosis is required to find a recommendation. However, in order to identify a suitable diagnosis for a group (i.e., for all group members), both *fairness* among group members and also the *satisfaction* of all group members must be taken into account. We propose an approach that takes into account all the aspects mentioned. In particular, our approach determines *socially-aware diagnoses* guided by aggregation functions in situations where the preferences of group members are inconsistent with the underlying constraint set. We show that the aspect of fairness (i.e., *Least Misery* aggregation function) plays a major role in the selection of high-involvement items. Furthermore, the prediction quality of our approach outperforms basic approaches such as *Breadth-First Search* and *Direct Diagnosis* (Atas et al., 2019a).

(*Q*7) *How can a constraint-based recommender system identify a recommendation which is similar to the preferences articulated by a user?*	We introduce two approaches to constraint-based recommendation that suggest suitable recommendations from a long list of recommendations to users. Our approaches can generate recommendations without identifying a diagnosis in cases where user requirements are inconsistent with the underlying constraint set. The first approach (referred to as the *soft relaxation-based approach*) identifies a similar recommendation by relaxing requirements which are in conflict with the underlying knowledge base. In the second approach (referred to as the *hard relaxation-based approach*), recommendations similar to the users requirements are identified by deleting the diagnosed user requirements. We test both approaches (*soft* and *hard relaxation-based approach*) with two different datasets and evaluate them with regard to their runtime performance and the degree of the similarity between the original requirements and the identified recommendations. The results show that both approaches are able to identify similar recommendations in an efficient way, even if the user requirements are inconsistent with the underlying knowledge base (Atas et al., 2019b).
(*Q*8) *How can requirement dependencies be identified automatically using supervised classification techniques?*	We introduce an intelligent approach for automatically identifying dependencies between software requirements of the type "requires" by using supervised classification techniques (Atas et al., 2018b). With the application of *Natural Language Processing (NLP) techniques*, our approach aims to identify requirement dependencies of the type "requires" by using the "title" and "description" of each requirement. First, the *NLP techniques* are applied to filter out irrelevant information (e.g., removing stop words). Subsequently, different classifiers such as *Naive Bayes, Linear SVM, k-Nearest Neighbors,* and *Random Forest* are trained and tested to identify the correct dependencies of type "requires". The results indicate that *Random Forest classifiers* correctly predict dependencies with an F_1 score of 0.82.

(*Q*9) *How can configuration knowledge be efficiently represented in the IoT domain?*	In this context, we investigate how to represent configuration knowledge easily and efficiently in the domain of *smart homes*. Usually, configuration problems in the *Internet of Things* (IoT) domain consist of hundreds of different components and constraints and representing this knowledge in a traditional way (e.g., constraint-based representation) is not easily manageable. As a result, there is an urgent need for a component-oriented knowledge representation that is easy to use and maintain. In order to tackle these open issues, we propose a knowledge representation described in *Answer Set Programming* (ASP) language which is an alternative to constraint-based knowledge representations and useful for large and complex configuration domains. In addition, we show that this logic-based approach is well-suited for a component-oriented representation of configuration tasks in the IoT domain (Felfernig et al., 2017b).

Table 1.1.: Overview of the contributions with regard to the research questions of this thesis.

1.4. Thesis Outline

The remainder of this thesis is structured as follows:

Chapter 2 gives an introduction to recommendation approaches for individual users and shows how to integrate these in group recommendation scenarios. These scenarios are designed on the basis of the following recommendation techniques: *collaborative filtering, content-based filtering, constraint-based including utility-based recommendation, critiquing-based,* and *hybrid recommendation*. In addition, basic strategies such as *aggregated predictions* and *aggregated models* are applied to aggregate the preferences of individual group members.

In **Chapter 3**, the selection of preference aggregation functions in dependence on the item domain is analyzed based on a dataset collected in a user study. In particular, item domains are separated based on their decision effort (*high/low-involvement item*). The following group aggregation functions are analyzed: *Average Voting, Least Misery, Most Pleasure, Minimal Group Distance, Ensemble Voting,* and *Multiplicative*.

Chapter 4 presents possibilities for increasing the amount of exchanged knowledge in group decision scenarios. The idea is to increase the frequency of information exchange, which will lead to an improvement of the corresponding decision quality. In addition, the impact of recommendation diversity on knowledge sharing among group members is analyzed and evaluated based on an empirical study conducted with groups of computer science students.

Chapter 5 provides an overview of *"Group Polarization Effects"* and shows the impact of these effects on the decision making behaviour of group members. The influence of these effects on decisions is analyzed in the following two dimensions: *risk analysis* and *cost estimation*. These effects are investigated particularly in situations where individual group members tend to make either risky or cautious decisions. Apart from analyzing scenarios in which group polarization effects occur, we also sketch how to avoid such effects.

Chapter 6 describes the application of constraint-based recommender systems for groups and how to apply *consensus-based* (Average Voting, Minimal Group Distance, Multiplicative, and Ensemble Voting) and *borderline* (Least Misery and Most Pleasure) preference aggregation functions to predict items for a group of users. This chapter analyzes the prediction capability of different aggregation functions in situations where no solution can be found for a given set of preferences (i.e., over-constrained). The prediction quality of aggregation functions is evaluated based on a dataset collected in a user study.

Chapter 7 presents a guided approach that determines *socially-aware diagnoses* (i.e., diagnoses suitable for the whole group of users) based on preference aggregation functions. For this purpose, we designed a user study, collected a dataset in a domain of high-involvement items (*digital cameras*) and synthesized the dataset from individual participants to generate groups of users. Finally, the prediction quality of these preference aggregation functions has been measured by analyzing the synthesized dataset.

Chapter 8 introduces two different approaches for providing recommendations which are similar to requirements articulated by users. In particular, we show how these similarity-aware constraint-based

approaches (i.e., recommender systems) can identify similar recommendations, even if the user requirements are inconsistent with the underlying knowledge base. Furthermore, we test our approaches with two real-world datasets and evaluate them with respect to runtime performance and degree of similarity between the original requirements and the identified recommendation.

Chapter 9 presents an intelligent approach for the automated identification of requirement dependencies. This approach analyzes the title and the description of each requirement and detects dependencies by using supervised classification techniques. Afterwards, we analyze our approach based on a dataset collected in a user study with different classifiers. Finally, *Naive Bayes, Linear SVM, k-Nearest Neighbors, and Random Forest* classifiers are evaluated according to their *precision, recall,* and F_1 scores.

In **Chapter 10**, configuration scenarios in the *Internet of Things* product domain are introduced. Since the size and complexity of configuration problems in the *Internet of Things* domain are rapidly increasing, we introduce a component-oriented knowledge representation, which is easier to maintain than the traditional knowledge representations (e.g., constraint-based representation). We show how the configuration knowledge in the domain of smart homes can be represented on the basis of *"Answer Set Programming"*. Moreover, we show that our logic-based approach is well-suited for a component-oriented representation of configuration tasks.

In **Chapter 11**, we conclude the thesis and give an outlook on future research issues.

Chapter 2

Algorithms for Group Recommendation

Parts of the contents of this chapter have been published in (Felfernig et al., 2018a). The author of this thesis provided parts of this chapter in terms of writing and literature research.

Abstract In this chapter, our aim is to show how group recommendation can be implemented on the basis of recommendation paradigms for individual users. Specifically, we focus on collaborative filtering, content-based filtering, constraint-based, critiquing-based, and hybrid recommendation. Throughout this chapter, we differentiate between (1) *aggregated predictions* and (2) *aggregated models* as basic strategies for aggregating the preferences of individual group members.

2.1. Introduction

There are many real-world scenarios where recommendations have to be made to groups. The main task in these scenarios is to generate relevant recommendations from the preferences (evaluations) of individual group members. As illustrated in Table 2.1, group recommendation approaches can be differentiated with regard to the following characteristics (Masthoff, 2011, 2015).

Preference Aggregation Strategy. In group recommender systems, there are two basic aggregation strategies (Jameson and Smyth, 2007). First, recommendations are determined for individual group members and then aggregated into a group recommendation.* Second, the preferences of individual users are aggregated into a *group profile* which is then used to determine a group recommendation. In this chapter, we show how both strategies can be applied with different recommendation algorithms.

Recommendation Algorithm. The recommendation logic of group recommenders is in many cases based on single user recommenders (collaborative filtering, content-based filtering, constraint-based, critiquing-based, and hybrid recommendation) (Felfernig et al., 2013a) combined with selected *aggregation functions* from *social choice theory* (Masthoff, 2011; Pennock et al., 2000). These functions will be discussed on the basis of examples from the travel domain.

Preferences Known Beforehand. Consider the example of single-shot recommendations determined on the basis of collaborative filtering. Some user preferences are already known from previous recommendation sessions, and so do not need to be determined in an iterative process. In contrast, conversational recom-

*One can also distinguish between the *aggregation of items* and the *aggregation of evaluations* (e.g., *ratings* in collaborative filtering) (Jameson and Smyth, 2007; Boratto et al., 2017) – in this chapter we will provide examples of both.

characteristic	description
Preference Aggregation Strategy	(1) determination of *items/ratings* for individual group members, thereafter aggregation of these items/ratings to a group recommendation, or (2) aggregation of the preferences of group members into a *group profile*, thereafter determination of a recommendation for the group.
Recommendation Algorithm	One of the recommendation algorithms (i.e., collaborative, content-based, constraint-based, critiquing-based, and hybrid).
Preferences Known Beforehand?	For example, in collaborative filtering, ratings are known beforehand (Baltrunas et al., 2010). In conversational approaches, preferences are constructed over time (Jameson et al., 2015).
Immediate Item Consumption?	Group recommenders can recommend (1) items that will be *consumed in the future* (e.g., holiday destinations as a basis for a final decision taken by (a) a responsible person or (b) a group on the basis of a discussion (Jameson et al., 2004)), or (2) items *consumed immediately* (e.g., songs (Masthoff, 2011)).
Active or Passive Group?	A group is *passive* if it does not actively influence the construction of a group profile (Baltrunas et al., 2010). *Active* groups negotiate the group profile (Jameson, 2004; Nguyen and Ricci, 2017).
Number of Recommended Items	A group recommender can focus on the recommendation of (1) a single item as is the case with travel destinations (Jameson, 2004) or (2) multiple items represented, for example, as a sequence (e.g., television items (Masthoff, 2011)).
Type of Preference Acquisition	Preferences can be acquired by interpreting, for example, the ratings of items or by engaging users in a preference construction process (Jameson et al., 2004).

Table 2.1.: Characteristics to classify group recommenders (Masthoff, 2011, 2015).

mender systems (McCarthy et al., 2006; Felfernig and Burke, 2008; Mahmood and Ricci, 2009; Chen et al., 2015; Christakopoulou et al., 2016; Nguyen, 2017) engage users in a dialog to elicit user preferences.

Immediate Item Consumption. On the one hand, a pragmatics of a recommendation can be that a group directly experiences the recommended items. For example, consider songs consumed by members of a fitness studio or commercials shown on public screens. On the other hand, recommendations are often interpreted as proposals without the items being experienced immediately.

Active or Passive Group. On the one hand, group profiles can be generated automatically if the preferences of the group members are known. On the other hand, especially when using constraint-based or critiquing-based recommenders, preferences are constructed (i.e., not known beforehand) and thus are adapted and extended within the scope of negotiation processes. The more intensively group models are discussed and negotiated, the higher the degree of group activity.

Number of Recommended Items. The output of a group recommender can be a single item (e.g., restaurant for a dinner or a movie), but also packages (e.g., travel packages), sequences (e.g., songs or travel plans), and even configurations (e.g., software release plans and cars).

Type of Preference Acquisition. Preferences can be collected *implicitly* (through observation, for example, of user's item consumption patterns) or *explicitly* by engaging users in a preference construction process. The latter is the case especially in conversational recommendation (McCarthy et al., 2006; Felfernig and Burke, 2008; Mahmood and Ricci, 2009; Chen et al., 2015; Christakopoulou et al., 2016).

2.2. Preference Aggregation Strategies

Independent of the way preferences are acquired from individual group members (see Chapter 5), a group recommendation is determined by aggregating these preferences in one way or another (Jameson and Smyth, 2007). In group recommender systems, the determination of recommendations depends on the chosen *preference aggregation strategy* (Yu et al., 2006; Jameson and Smyth, 2007; Baltrunas et al., 2010; Garcia-Molina et al., 2011; Kompan and Bielikova, 2014; Marquez and Ziegler, 2015; Boratto et al., 2017).

There are two aggregation strategies (see Figure 2.1): (1) *aggregating recommended items* (or *evaluations*) that were generated separately for each user profile $up(u_i)$ and (2) *aggregating individual user profiles* $up(u_i)$ into a group profile gp. In the first case, *the recommendation step precedes the aggregation step* – item evaluations or items recommended to individual group members are *aggregated* into a corresponding group recommendation. In the second case, *the aggregation step precedes the recommendation step* – group profiles aggregated from individual user profiles are the basis for determining a group recommendation. Following the discussions in (Jameson and Smyth, 2007; Berkovsky and Freyne, 2010), we denote the first aggregation strategy as *aggregated predictions* and the second one as *aggregated models* (see Figure 2.1).

Aggregated Predictions. There are two basic approaches to aggregate predictions. *First*, recommendations (items) determined for individual group members can be merged. This approach can be used if a *a set of candidate solutions* should be presented and the group members are in charge of selecting one out of the candidate items. In this context, specific items which are not very appealing for some group members are not filtered out. Group members play an important role in the decision making process, since *no ranking* of the individual candidate items is provided. *Second*, group-member-specific predictions for candidate items are aggregated. The outcome of this approach is a *ranking of candidate items*.

Aggregated Models. Instead of aggregating recommendations for individual users, this approach constructs a *group preference model* (*group profile*) that is then used for determining recommendations. This is especially useful in scenarios where group members should have the opportunity to *analyze, negotiate, and adapt the preferences of the group* (Jameson and Smyth, 2007). Another advantage of applying group preference models is that the *privacy concerns* of users can be alleviated, since there is no specific need to record and maintain individual user profiles.

Although studies exist that compare the predictive quality of the two basic aggregation approaches (*aggregated predictions* and *aggregated models*) (Baltrunas et al., 2010; Berkovsky and Freyne, 2010; DePessemier et al., 2013; Boratto and Carta, 2015), more in-depth comparisons are needed that also focus on specific group properties such as size, homogeneity (e.g., similarity between group members can have

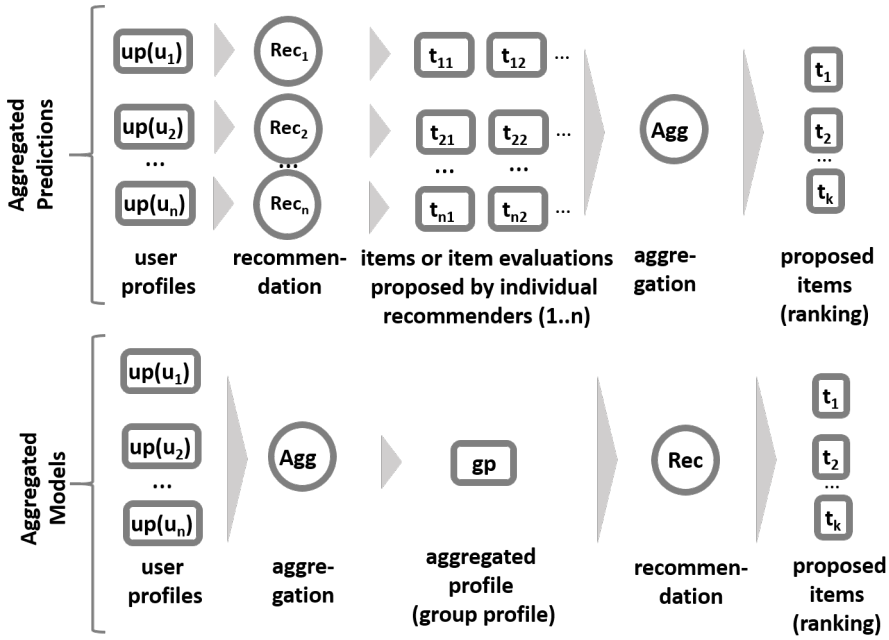

Figure 2.1.: Two basic aggregation strategies in group recommendation: (1) recommendation based on *single user profiles* with a downstream aggregation of items (or evaluations/ratings) recommended to group members/users (*aggregated predictions*) and (2) recommendation based on *aggregated models* (*group profiles*).

a negative impact on the decision quality), the item domain (e.g., high-involvement vs. low-involvement items (Felfernig et al., 2017a)), and also the ways in which individual and group rating behavior differs (Sacharidis, 2017). After introducing a couple of *social choice based preference aggregation functions* that help to implement aggregated predictions and aggregated models, we show how preference aggregation can be implemented in the context of *collaborative-* and *content-based filtering* as well as *constraint-based*, *critiquing-based*, and *hybrid recommendation*.

2.3. Social Choice based Preference Aggregation Functions

A major issue in all of the mentioned group recommendation scenarios is how to adapt to the group as a whole, given information about the individual preferences of group members (Arrow, 1950; Masthoff, 2011). As there is *no optimal way* to aggregate recommendation lists (Arrow, 1950), corresponding approximations (in the following denoted as *aggregation functions*) have to be used to come up with a recommendation that takes into account 'as far as possible' the individual preferences of group members. As mentioned in (Senot et al., 2010; Masthoff, 2015), the aggregation functions can be categorized into *majority-based* (*M*), *consensus-based* (*C*), and *borderline* (*B*). Table 2.2 provides an overview of different kinds of *aggregation functions* taken from *social choice theory*[†] (Masthoff, 2004; Chevaleyre et al., 2007; Masthoff, 2011; Senot et al., 2011) and their categorization into one of the three mentioned categories (*M*, *C*, and *B*).

[†]Also denoted as *group decision making*.

Majority-based aggregation functions (M) represent aggregation mechanisms that focus on those items which are the *most popular* (Masthoff, 2004; Senot et al., 2011). Examples of majority-based functions are *Plurality Voting* (PLU) (winner is the item with the highest number of *votes*), *Borda Count* (BRC) (winner is the item with the best *total ranking score* where each item rank[‡] is associated with a score 0 .. $\#items-1$), and *Copeland Rule* (COP) (winner is the item that outperforms other items in terms of pairwise *evaluation*[§] comparison) (see Table 2.3). Equal evaluations in BRC are handled as follows: in the example of Table 2.3, user u_2 provided the rating 2.5 for t_2 and t_3; both items receive the same *score* which is $\frac{0+1}{2} = 0.5$.

aggregation strategy	description	recommendation
Additive Utilitarian (ADD) [C]	sum of item-specific evaluations	$\underset{(t \in I)}{argmax}(\Sigma_{u \in G} eval(u,t))$
Approval Voting (APP) [M]	number of item-specific evaluations above an approval threshold	$\underset{(t \in I)}{argmax}(\lvert\{u \in G : eval(u,t) \geq threshold\}\rvert)$
Average (AVG) [C]	average of item-specific evaluations	$\underset{(t \in I)}{argmax}(\frac{\Sigma_{u \in G} eval(u,t)}{\lvert G \rvert})$
Average without Misery (AVM) [C]	average of item-specific evaluations (if all evaluations are above a defined threshold)	$\underset{(t \in I : \nexists u \in G \lvert eval(u,t) \leq threshold)}{argmax}(\frac{\Sigma_{u \in G} rating(u,t)}{\lvert G \rvert})$
Borda Count (BRC) [M]	sum of item-specific scores derived from item ranking	$\underset{(t \in I)}{argmax}(\Sigma_{u \in G} score(u,t))$
Copeland Rule (COP) [M]	number wins (w) - number losses (l) in pair-wise evaluation comparison	$\underset{(t \in I)}{argmax}(\lvert w(t, I-\{t\})\rvert - \lvert l(t, I-\{t\})\rvert)$
Fairness (FAI) [C]	item ranking as if individuals ($u \in G$) choose them one after the other	$\underset{(t \in I)}{argmax}(eval(u,t))$ [in each iteration]
Least Misery (LMS) [B]	minimum item-specific evaluation	$\underset{(t \in I)}{argmax}(mineval(t))$
Majority Voting (MAJ) [B]	majority of evaluation values per item	$\underset{(t \in I)}{argmax}(majorityeval(t))$
Most Pleasure (MPL) [B]	maximum item-specific evaluation	$\underset{(t \in I)}{argmax}(maxeval(t))$
Most Respected Person (MRP) [B]	item-evaluations of most respected user	$\underset{(t \in I)}{argmax}(eval(u_{mrp},t))$
Multiplicative (MUL) [C]	multiplication of item-specific evaluations	$\underset{(t \in I)}{argmax}(\Pi_{u \in G} eval(u,t))$
Plurality Voting (PLU) [M]	item with the highest #votes from $u \in G$	$\underset{(t \in I)}{argmax}(votings(t))$ [in each iteration]

[‡]The highest rank is assumed to be 1. For example, in collaborative filtering it is associated with the highest rating. The highest rank is associated with the score *#items-1*.

[§]For example, when using collaborative filtering, evaluations are denoted as *ratings*.

item	votes			*PLU*	evaluations (scores)			*BRC*	evaluations			COP index			*COP*
	u_1	u_2	u_3		u_1	u_2	u_3		u_1	u_2	u_3	t_1	t_2	t_3	
t_1	1	1	0	2 $\surd$	5.0(2)	4.5(2)	3.5(1)	5 $\surd$	5.0	4.5	3.5	0	+	+	2$\surd$
t_2	0	0	1	1	3.0(0)	2.5(0.5)	4.0(2)	2.5	3.0	2.5	4.0	-	0	0	-1
t_3	0	0	0	0	3.5(1)	2.5(0.5)	1.5(0)	1.5	3.5	2.5	1.5	-	0	0	-1

Table 2.3.: Examples of *majority-based* aggregation: Plurality Voting (PLU), Borda Count (BRC), and Copeland Rule (COP, "+" indicates a win, "-" a loss, and "0" a tie). $\surd$ denotes the item t_i with the best evaluation, i.e., the *recommendation*.

Table 2.2.: Basic *aggregation functions* for group recommender systems (Chevaleyre et al., 2007; Levin and Nalebuff, 1995; Masthoff, 2011, 2015; Senot et al., 2010) where *argmax* is assumed to return a recommended item. Tie breaking rules such as *random selection* can be applied. *M*, *C*, and *B* denote the aggregation categories *majority-based*, *consensus-based*, and *borderline*; *u* represents a *user* (*group member*), *G* a *group*, *t* an *item*, and *I* a set of *items*.

When comparing the items t_1 and t_2 in Table 2.3, t_1 outperforms t_2 two times and looses once in terms of user evaluations ($u_1 : 5.0$ vs. $u_2 : 3.0$, $u_1 : 4.5$ vs. $u_2 : 2.5$, and $u_1 : 3.5$ vs. $u_2 : 4.0$) which results in a win ("+") 2:1. Comparing items t_2 and t_3 results in a tie 1:1 which is indicated by "0" in Table 2.3. Such an evaluation has to be performed for each item in order to determine a winner on the basis of COP (see the *rhs* of Table 2.3). A further majority-based aggregation function is *Approval Voting* (APP) that recommends items with the highest number of supporting users. In this context, support is measured in terms of the number of item evaluations above a defined threshold.

Consensus-based functions (C)[¶] represent aggregation mechanisms that take into account the preferences of *all group members* (Senot et al., 2011). Examples are *Additive Utilitarian* (ADD) (winner is the item with the maximum sum of user-individual evaluations), *Average* (AVG) (winner is the item with the maximum average of the user-individual evaluations – in the line of ADD[‖], the function causes problems in the context of larger groups since the opinions of individuals count less), and *Multiplicative* (MUL) (winner is the item with the maximum product of the user-individual evaluations) (see Table 2.4). Further majority-based aggregation functions are *Average without Misery* (AVM) that recommends the average evaluation for items that do not have individual ratings below a defined threshold and *Fairness* (FAI) which ranks items as if individuals are choosing them in turn (Masthoff, 2011).
Borderline functions (B) represent aggregation mechanisms that take into account only *a subset of the user preferences* (Senot et al., 2011). Examples of borderline functions are *Least Misery* (LMS) (winner is the item with the highest of all lowest evaluations given to items – when using this function, items may be selected that nobody hates but also nobody really likes; furthermore, there is the danger that a minority dictates the group (especially in settings involving larger groups) (Masthoff, 2004)), *Most Pleasure* (MPL) (winner is the item with the highest of all individual evaluations – items may be selected that only a few persons really like)[**], and *Majority Voting* (MAJ) (item with the highest number of all evaluations representing

[¶] Also denoted as *democratic functions*.

[‖] ADD and AVG result in the same rankings.

[**] Variants thereof can be considered (Masthoff, 2004), for example, *most pleasure without misery* where only items are considered that do not have evaluations below a predefined threshold.

item	evaluations			*ADD*	*AVG*	*MUL*
	u_1	u_2	u_3			
t_1	5	2	2	9	3	20
t_2	3	3	4	10$\surd$	3.3 $\surd$	36 $\surd$
t_3	2	3	2	7	2.3	12

Table 2.4.: Examples of *consensus-based* aggregation: Additive Utilitarian (ADD), Average (AVG), and Multiplicative (MUL).

item	evaluations			*LMS*	*MPL*	*MAJ*
	u_1	u_2	u_3			
t_1	5	2	2	2	5 $\surd$	2
t_2	3	3	4	3$\surd$	4	3 $\surd$
t_3	2	3	2	2	3	2

Table 2.5.: Examples of *Borderline* aggregation: Least Misery (LMS), Most Pleasure (MPL), and Majority Voting (MAJ).

the majority of item-specific evaluations) (see Table 2.5). A further borderline aggregation function is *Most Respected Person* (*MRP*) that recommends a rating (evaluation) proposed by the most respected individual.

The following discussions of group recommendation approaches will be based on a set of example items from the travel domain. Using these items, we will show how different recommendation approaches can determine group recommendations with *aggregated models* and *aggregated predictions*.

2.4. Collaborative Filtering for Groups

Collaborative filtering (CF) (Konstan et al., 1997; Linden et al., 2003) is based on the idea of recommending items that are derived from the preferences of *nearest neighbors*, i.e., users with preferences similar to those of the current user. In the following, we show how *aggregated predictions* and *aggregated models* can be applied to CF for groups.

Aggregated Predictions. When applying the *aggregated predictions* strategy in combination with collaborative filtering, ratings are determined for individual users and then aggregated into a recommendation for the group (see Figure 2.2).
Following this approach, for each group member (and corresponding recommender) i and each item j not rated by this group member, a rating prediction $\hat{r}_{ij}$ is determined (Berkovsky et al., 2010). For simplicity, we assume that the items $\{t_1,..,t_{10}\}$ in Table 2.6 have not been previously consumed by the group members, i.e., the rating has been proposed by a collaborative filtering algorithm.[††] Thereafter, these predictions are aggregated on the basis of different aggregation functions (see Table 2.2). In the following example, we assume that some variant of collaborative filtering (Ekstrand et al., 2011; Quijano-Sánchez et al., 2013; Ghazarian and Nematbakhsh, 2015) has already been applied to predict ratings (e.g., a matrix factorization approach (Ortega et al., 2016; Sacharidis, 2017) can be applied to infer $user \times item$ rating

[††]Item predictions for individual users can be based on collaborative recommendation approaches.

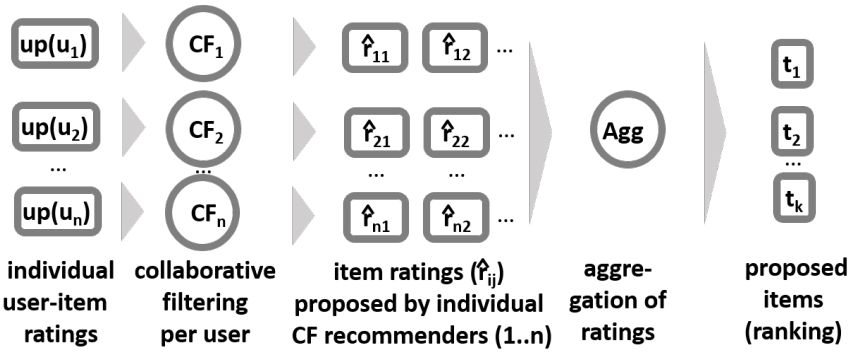

Figure 2.2.: Collaborative filtering for groups based on *aggregated predictions* (*ratings*). $\hat{r}_{ij}$ is the rating prediction for item j proposed by recommender i ($i = 1..n$).

tables as shown in Table 2.6).

item	name	rating predictions $\hat{r}_{ij}$ (scores)					aggregation		
		u_1	u_2	u_3	u_4	u_5	AVG	BRC	LMS
t_1	Vienna	5.0(9)	3.5(2)	1.0(0)	4.5(7)	5.0(9)	3.8	27	1.0
t_2	Yellowstone	2.5(0)	4.0(4)	3.0(3)	2.0(0)	1.1(0)	2.5	7	1.1
t_3	New York	4.9(8)	3.8(3)	4.0(7)	3.3(4)	4.0(5)	4.0	27	3.3$\surd$
t_4	Blue Mountains	3.1(2)	5.0(9)	4.2(8)	2.4(1)	4.4(8)	3.8	28	2.4
t_5	London	4.0(4)	4.3(7)	3.3(5)	4.1(6)	2.9(3)	3.7	25	2.9
t_6	Beijing	4.5(6)	4.1(5)	5.0(9)	3.2(3)	4.2(6)	4.2$\surd$	29$\surd$	3.2
t_7	Cape Town	4.2(5)	4.2(6)	3.4(6)	3.1(2)	3.8(4)	3.7	23	3.1
t_8	Yosemity	3.4(3)	2.6(0)	1.6(1)	5.0(9)	2.4(2)	3.0	15	1.6
t_9	Paris	4.7(7)	3.1(1)	2.7(2)	3.6(5)	2.2(1)	3.3	16	2.2
t_{10}	Pittsburgh	2.6(1)	4.5(8)	3.1(4)	4.6(8)	4.3(7)	3.8	28	2.6

Table 2.6.: Rating predictions and corresponding *scores* (scores are used by *BRC*). Recommendations are derived on the basis of aggregation functions (*AVG*, *BRC*, *LMS*). The $\surd$ symbol indicates the item with the best evaluation.

The result of the aggregation step is a *ranking of candidate items*. In our example, the majority of aggregation functions recommends the item t_6. An alternative to the aggregation of ratings is to *aggregate predicted items* where *items* determined by individual recommenders are aggregated into a group recommendation (see Figure 2.3).

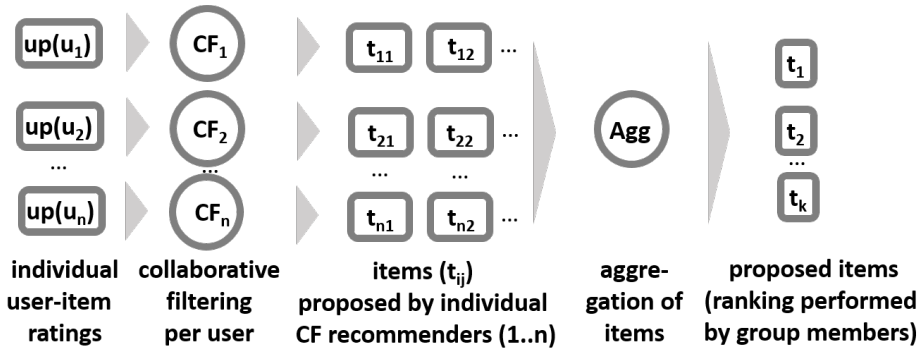

Figure 2.3.: Collaborative filtering for groups based on *aggregated predictions* (*items*).

Following this approach, items with the highest predicted rating for a specific user are considered as part of the recommendation. If we want to generate a recommendation consisting of, for example, at most 10 items, the two top-rated items (upper bound) in each group member specific recommendation can be included in the group recommendation. In the example shown in Table 2.6, $\{t_1, t_3\}$ are the two top-rated items of user u_1, $\{t_4, t_{10}\}$ are chosen for user u_2, $\{t_4, t_6\}$ for user u_3, $\{t_8, t_{10}\}$ for user u_4, and $\{t_1, t_4\}$ for user u_5. The union of these group member individual recommendations is $\{t_1, t_3, t_4, t_6, t_8, t_{10}\}$ which represents the group recommendation – in this context, group members are in charge of item ranking. This way of constructing a group recommendation is similar to the idea of the *Fairness* (FAI) aggregation function (see Table 2.2).

Aggregated Models. When using this aggregation approach, ratings of individual users are aggregated into a group profile gp (see Figure 2.4). Based on the group profile (gp), collaborative filtering determines a ranking for each candidate item.

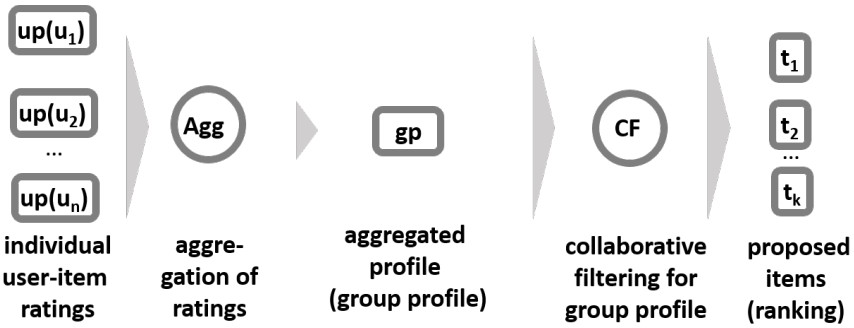

Figure 2.4.: Collaborative filtering for groups based on *aggregated models*.

In the aggregated models approach, the group is represented by a *group profile* (gp) that includes item-specific evaluations (ratings) derived through aggregation functions applied to the item ratings of individual group members. Often, the aggregation is based on a weighted average function (see, e.g., (Berkovsky et al., 2010)), however, the aggregation functions mentioned in Table 2.2 can be considered alternatives. Following the aggregated models strategy, collaborative filtering is applied to individual group profiles, i.e., for a given group profile (gp), similar group profiles (k nearest neighbors k-NN)[‡‡] are retrieved and used for determining a recommendation. In our example, the item t_2 (*Yellowstone*) is *not* known to the current group gp but received the highest ratings from the nearest neighbor groups gx and gy (see Table 2.7) which makes it a recommendation candidate for gp.

item	name	gp	$gx \in NN$	$gy \in NN$	recommended ratings
t_1	Vienna	5.0	5.0	4	-
t_2	Yellowstone	-	4.0	4.5	$4.49\surd$
t_3	New York	4.0	3.0	3.5	-
t_4	Blue Mountains	-	4.5	4	4.44
t_5	London	4.0	3.9	3.5	-
t_6	Beijing	-	3.5	3	3.44
t_7	Cape Town	-	4.7	3	3.99
t_8	Yosemity	3.0	3.8	3.2	-
t_9	Paris	4.0	3.9	2.9	-
t_{10}	Pittsburgh	-	5.0	3.3	4.28
average		4.0	4.13	3.5	-

Table 2.7.: Applying collaborative filtering (CF) to a group profile gp (gp-ratings have no relationships to earlier examples). The $\surd$ symbol indicates the item with the best CF-based evaluation.

The *similarity* between the group profile gp and another group profile gx (the nearest neighbor) can be determined, for example, using the Pearson correlation coefficient. Formula 2.1 is an adapted version that determines the similarity between a group profile and the profiles of other groups. In this context, TD_c represents the set of items that have been rated by both groups (gp and gx), r_{gx,t_i} is the rating of group gx for item t_i, and $\overline{r_{gx}}$ is the average rating of group gx.

$$similarity(gp,gx) = \frac{\sum_{t_i \in TD_c}(r_{gp,t_i} - \overline{r_{gp}}) \times (r_{gx,t_i} - \overline{r_{gx}})}{\sqrt{\sum_{t_i \in TD_c}(r_{gp,t_i} - \overline{r_{gp}})^2} \times \sqrt{\sum_{t_i \in TD_c}(r_{gx,t_i} - \overline{r_{gx}})^2}} \tag{2.1}$$

The information about groups with a similar rating behavior (i.e., nearest neighbors NN) compared to the current group gp is the basis for predicting the rating of gp for an *item* t that has not been rated by members of gp (see Formula 2.2).

$$prediction(gp,t) = \hat{r}(gp,t) = \overline{r_{gp}} + \frac{\sum_{gj \in NN} similarity(gp,gj) \times (r_{gj,t} - \overline{r_{gj}})}{\sum_{gj \in NN} similarity(gp,gj)} \tag{2.2}$$

[‡‡] In our example, we assume $k = 2$.

Recommendations can also be determined on the basis of *ensemble voting* (Stettinger and Felfernig, 2014): each aggregation function can represent a vote. The more such votes an item receives, the higher is it's relevance for the group. In our running example, item t_6 is regarded as favorite item since it received the best evaluation by the majority of the used aggregation functions (see the *aggregated predictions* example in Table 2.6).

2.5. Content-based Filtering for Groups

Content-based filtering (CBF) is based on the idea of recommending new items with *categories*[§§] similar to those preferred by the current user. Categories preferred by a user (group member) are stored in a user profile; these categories are derived from descriptions of items already consumed by the user.
Aggregated Predictions. When using this aggregation strategy, group member individual content-based recommenders determine the similarity between (a) items not consumed by him/her and (b) his/her user profile.[¶¶] The identified item similarities (or items) are then aggregated and thus form the basis of a group recommendation (see Figure 2.5).

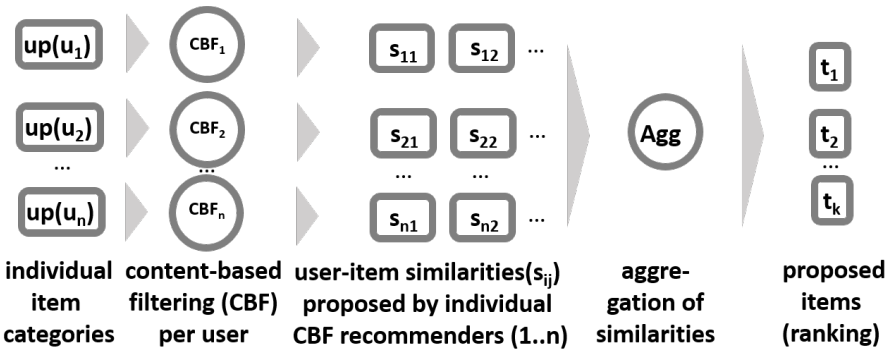

Figure 2.5.: Content-based filtering for groups based on *aggregated predictions*. Similarity s_{ij} denotes the similarity between user i and item j determined by recommender i ($i = 1..n$).

Table 2.9 depicts example profiles of group members $u_1..u_5$. For each of these profiles, the similarity to the items included in Table 2.8 is determined (we assume that these items have not been consumed/evaluated by the group members). These similarity values are the basis for a group recommendation (see Table 2.10).

The user-item similarities of Table 2.10 are calculated by a content-based recommender (see *similarity* metrics in (McSherry, 2004)). The calculation is based on the item categories included in Table 2.9, i.e., *beach*, *city tours*, *nature*, and *entertainment*. For example, $similarity(u_1,t_2) = \frac{2*|categories(u_1) \cap categories(t_2)|}{|categories(u_1)|+|categories(t_2)|}$ $= \frac{2}{3} = 0.66$.

On the basis of a user $\times$ item similarity matrix, aggregation functions can determine a group recommendation. An alternative to the aggregation of similarities is to *aggregate items* proposed by individual content-based recommenders. If we want to generate a recommendation consisting of, for example, at most 5 items (upper bound), the highest rated item of each group member can be included in the group recommendation.

[§§] Alternatively, *keywords* extracted from item descriptions.
[¶¶] The determination of *user* $\times$ *item similarities* can be based on content-based recommendation approaches.

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 2.8.: 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*).

category	individual item categories				
	u_1	u_2	u_3	u_4	u_5
beach	x	x	x	x	x
city tours	-	x	-	x	-
nature	x	-	x	-	x
entertainment	-	-	-	-	-

Table 2.9.: Example profiles of group members (preferences regarding travel destinations). If a group member u_i likes a category, this is denoted with 'x'.

In our example depicted in Table 2.10, $\{t_2\}$ is among the highest rated items of user u_1 (the other three are excluded due to the user-specific limit of one item), t_7 can be selected for user u_2, t_4 for user u_3, t_{10} for user u_4, and t_2 for user u_5. The group recommendation includes all of these items: $\{t_2, t_4, t_7, t_{10}\}$.

Aggregated Models. When using this strategy, preferred categories of individual users are integrated into a group profile gp. Thereafter, content-based filtering determines recommendations by calculating the similarities between gp and candidate items (items not consumed by the group – see Figure 2.6).

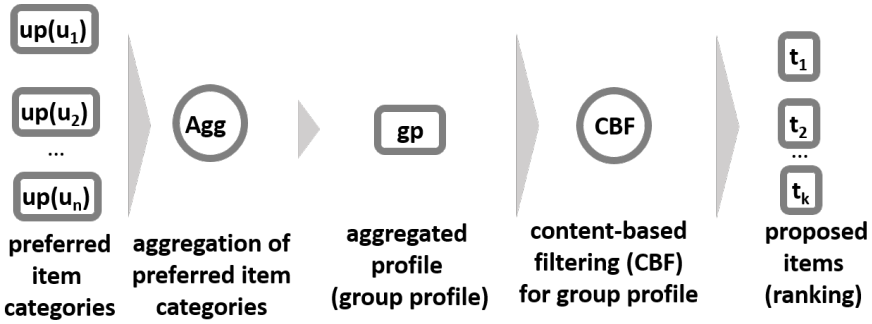

Figure 2.6.: Content-based filtering for groups based on *aggregated models*.

item	name	user-item similarities (scores)					aggregation		
		u_1	u_2	u_3	u_4	u_5	AVG	BRC	LMS
t_1	Vienna	0(2.5)	0.5(5)	0(2.5)	0.5(5)	0(2.5)	0.2	17.5	0
t_2	Yellowstone	0.66(7.5)	0(1)	0.66(7.5)	0(1)	0.66(7.5)	0.4	24.5	0
t_3	New York	0(2.5)	0.5(5)	0(2.5)	0.5(5)	0(2.5)	0.2	17.5	0
t_4	Blue Mountains	0.66(7.5)	0(1)	0.66(7.5)	0(1)	0.66(7.5)	0.4	24.5	0
t_5	London	0(2.5)	0.5(5)	0(2.5)	0.5(5)	0(2.5)	0.2	17.5	0
t_6	Beijing	0(2.5)	0.5(5)	0(2.5)	0.5(5)	0(2.5)	0.2	17.5	0
t_7	Cape Town	0.66(7.5)	0.66(8.5)	0.66(7.5)	0.66(8.5)	0.66(7.5)	$0.66\surd$	$39.5\surd$	$0.66\surd$
t_8	Yosemity	0.66(7.5)	0(1)	0.66(7.5)	0(1)	0.66(7.5)	0.4	24.5	0
t_9	Paris	0(2.5)	0.5(5)	0(2.5)	0.5(5)	0(2.5)	0.2	17.5	0
t_{10}	Pittsburgh	0(2.5)	0.66(8.5)	0(2.5)	0.66(8.5)	0(2.5)	0.26	24.5	0

Table 2.10.: User $\times$ item similarities (and corresponding *scores* used by *BRC*) as input for *AVG*, *BRC*, *LMS* to derive a group recommendation. The $\surd$ symbol indicates the item with the best evaluation.

In our example (see Table 2.11), the derived group profile is represented by the union of the categories stored in the individual user profiles. Items are recommended that are similar to the categories in the group profile and have not been consumed by group members. In our example, the derived group profile gp entails the categories *Beach*, *City Tours*, and *Nature*.

category	individual item categories					gp
	u_1	u_2	u_3	u_4	u_5	
beach	x	x	x	x	x	x
city tours	-	x	-	x	-	x
nature	x	-	x	-	x	x
entertainment	-	-	-	-	-	-

Table 2.11.: Aggregation of preferences (categories) of group members into a group profile gp.

The similarity between the group profile gp and candidate items can be determined using Formula 2.3. The similarities between gp and items t_i (taken from our example itemset shown in Table 2.8) are determined by comparing the categories *beach*, *citytours*, *nature*, and *entertainment* (see Table 2.12). For example, $similarity(gp,t_1) = \frac{2*|categories(gp) \cap categories(t_1)|}{|categories(gp)|+|categories(t_1)|} = \frac{2}{5} = 0.4$. In this context, we assume that the items of Table 2.12 have not been consumed by the group.

$$similarity(gp,item) = \frac{2*|categories(gp) \cap categories(item)|}{|categories(gp)| + |categories(item)|} \tag{2.3}$$

item	name	similarity(gp, t_i)
t_1	Vienna	$\frac{2}{5} = 0.4$
t_2	Yellowstone	$\frac{2}{4} = 0.5$
t_3	New York	$\frac{2}{5} = 0.4$
t_4	Blue Mountains	$\frac{2}{4} = 0.5$
t_5	London	$\frac{2}{5} = 0.4$
t_6	Beijing	$\frac{2}{5} = 0.4$
t_7	Cape Town	$\frac{6}{7} = 0.86$ $\surd$
t_8	Yosemity	$\frac{2}{4} = 0.5$
t_9	Paris	$\frac{2}{5} = 0.4$
t_{10}	Pittsburgh	$\frac{2}{4} = 0.5$

Table 2.12.: Applying content-based filtering (CBF) to a group profile gp (see Table 2.11). The $\surd$ symbol indicates the item with the best evaluation determined by CBF.

2.6. Constraint-based Recommendation for Groups

Taking into account groups in constraint-based recommendation (Felfernig and Burke, 2008) requires the extension of recommendation task for individuals.

Definition (Recommendation Task for Groups). A recommendation task for groups can be defined by the tuple (G, $R = R_1 \cup .. \cup R_m$, I) where $G = \{u_1, u_2, .., u_m\}$ represents a group of users, $R_j = \{r_{1j}, r_{2j}, ..., r_{nj}\}$ represents a set of requirements (r_{ij} denotes the requirement i of group member j), and $I = \{t_1, .., t_k\}$ represents a set of items. The goal is to identify items in I which fulfill all requirements in R. A solution for a recommendation task can be defined as follows.
Definition (Recommendation Task for Groups – Solution). A solution for a recommendation task for groups (G, 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, R represents requirements defined by group members, and I represents a collection of items.
In group recommendation settings, each group member should specify his/her *requirements* (in our example, these are hard constraints related to season and topics) and *preferences* (*weights* or *soft constraints*) with regard to a set of *interest dimensions* (in our example, *security*, *attractiveness*, and *crowdedness*) – see Table 2.13. Requirements are constraints that are used to pre-select items, preferences specify weights that are used to rank the pre-selected items.

user	requirements		preferences (weights)		
	season	*topics*	*security*	*attractiveness*	*crowdedness*
u_1	r_{11} :spring	-	0.5	0.4	0.1
u_2	r_{12} :spring	r_{22} :citytours	0.2	0.7	0.1
u_3	-	r_{13} :entertainment	0.3	0.3	0.4
u_4	r_{14} :spring	-	0.6	0.2	0.2
u_5	-	r_{15} :citytours	0.1	0.8	0.1

Table 2.13.: User-specific *requirements* and *preferences* (*weights*).

In both scenarios, i.e., *aggregated predictions* and *aggregated models*, group members have to define their requirements and preferences.

Aggregated Predictions. We will first show how to handle aggregated predictions in constraint-based recommendation for groups (see Figure 2.7).

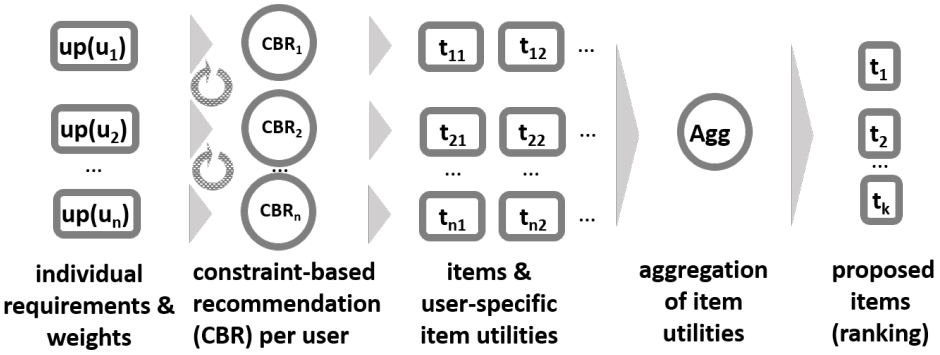

Figure 2.7.: Constraint-based recommendation for groups based on *aggregated predictions*. User preferences are constructed iteratively (conversational recommendation approach). Item t_{ij} represents item j (including corresponding item utilities) determined by recommender i.

A constraint-based recommender derives user-specific recommendations (items and user-specific item utilities) on the basis of a set of requirements and preferences. Item utilities for specific group members can be determined with multi-attribute utility theory (MAUT) (Winterfeldt and Edwards, 1986; Yu et al., 2006) (see Formula 2.4). For example, on the basis of the user requirements defined in Table 2.13 and the example itemset of Table 2.14, the utility of item t_1 for user u_1 can be determined as follows: $utility(u_1,t_1) = \sum_{d \in Dimensions} contribution(t_1,d) \times weight(u_1,d) = contribution(t_1,security) \times weight(u_1,security) + contribution(t_1,attractiveness) \times weight(u_1,attractiveness) + contribution(t_1,crowdedness) \times weight(u_1,crowdedness)$ = 5.0 × 0.5 + 5.0 × 0.4 + 2.0 × 0.1 = 2.5 + 2.0 + 0.2 = 4.7. These user-specific item utilities are aggregated into a group recommendation (see Table 2.15).

$$utility(u_a,item) = \sum_{d \in Dimensions} contribution(item,d) \times weight(u_a,d) \tag{2.4}$$

If an entry of item t_i in *user-specific item utilities* in Table 2.15 > 0, this indicates that the item t_i fulfills all requirements of the corresponding group member. In contrast, table entries $= 0$ are used to indicate that an item does not completely fulfill the requirements of a group member. For example, the requirements of u_2 ($\{r_{12}, r_{22}\}$) are not completely fulfilled by t_2 ($r_{22} : topics = citytours$ is not supported). Even if an item does not completely fulfill the requirements of some users, it could be recommended. The lower the number of users with completely fulfilled requirements with regard to a specific item t_i, the lower the probability that t_i will be recommended. A set of individual user requirements can also be inconsistent with the effect that no fitting item can be identified. In such a case, diagnosis methods can help to guide the user out of the *no solution could be found dilemma* (Felfernig et al., 2009b).[***]

[***]Issues related to conflict resolution will be discussed at the end of this section.

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 2.14.: 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).

Also in constraint-based recommendation, an alternative to the aggregation of user $\times$ item utilities (Table 2.15) is to *aggregate items* proposed by individual recommenders. If we want to generate a recommendation based on the *Fairness* (FAI) aggregation strategy and 5 is the upper bound for the number of proposed items, each group member would choose his/her favorite item (not already selected by another group member). In the example shown in Table 2.15, t_1 has the highest utility for user u_1, it also has the highest utility for user u_2, however, since u_1 already selected t_1, u_2 has to identify a different one, which is now t_3. Furthermore, we assume that u_3 selects t_9, u_4 selects t_8, and user u_5 selects t_5. The group recommendation resulting from this aggregation step is $\{t_1, t_3, t_5, t_8, t_9\}$.

Aggregated Models. Another possibility of determining recommendations for groups in constraint-based recommendation scenarios is to first aggregate individual user preferences (Jameson, 2004) (requirements and weights related to interest dimensions) into a *group profile gp* and then to determine recommendations (see Figure 2.8).

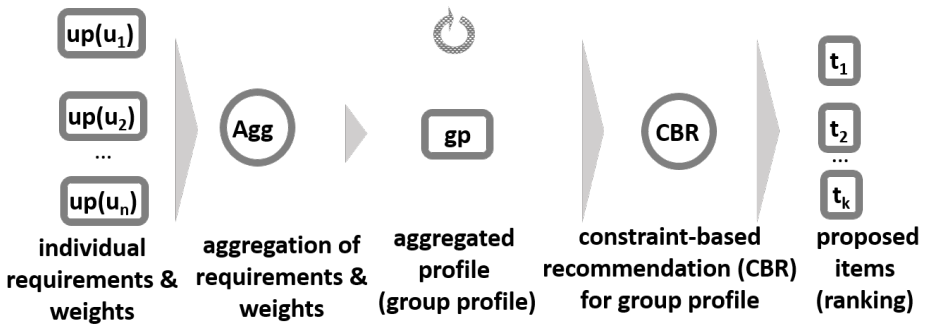

Figure 2.8.: Constraint-based recommendation for groups based on *aggregated models*. Group preferences are constructed iteratively (conversational recommendation).

item	item contribution			user-specific item utilities (scores)					aggregation		
	secur.	*attr.*	*crowd.*	u_1	u_2	u_3	u_4	u_5	AVG	BRC	LMS
t_1	5.0	5.0	2.0	4.7(9)	4.7(9)	3.8(9)	4.4(9)	4.7(9)	4.46$\surd$	45.0$\surd$	3.8$\surd$
t_2	4.0	4.0	4.0	4(7.5)	0.0	0.0	4(7)	0.0	1.6	14.5	0.0
t_3	3.0	5.0	1.0	3.6(4.5)	4.2(7.5)	2.8(7.5)	3(4)	4.4(7.5)	3.6	31.0	2.8
t_4	4.0	3.0	5.0	3.7(6)	0.0	0.0	4(7)	0.0	1.54	13.0	0.0
t_5	3.0	4.0	1.0	3.2(3)	3.5(6)	2.5(6)	2.8(2)	3.6(6)	3.12	23.0	2.5
t_6	3.0	3.0	1.0	2.8(1)	2.8(3.5)	2.2(5)	2.6(1)	2.8(3)	2.64	13.5	2.2
t_7	2.0	3.0	3.0	2.5(0)	2.8(3.5)	0.0	2.4(0)	2.9(4)	2.12	7.5	0.0
t_8	4.0	4.0	4.0	4(7.5)	0.0	0.0	4(7)	0.0	1.6	14.5	0.0
t_9	3.0	5.0	1.0	3.6(4.5)	4.2(7.5)	2.8(7.5)	3(4)	4.4(7.5)	3.6	31.0	2.8
t_{10}	3.0	3.0	3.0	3(2)	3(5)	0.0	3(4)	3(5)	2.4	16.0	3.0

Table 2.15.: User-specific item utilities (and corresponding *scores* used by *BRC*) with regard to *security*, *attractiveness*, and *crowdedness* determined by utility analysis. The $\surd$ symbol indicates the item with the best evaluation.

The construction of a group profile gp is sketched in Table 2.16. Beside aggregating the user requirements $R = \{r_{11}, r_{12}, r_{22}, r_{13}, r_{14}, r_{15}\}$, we also have to aggregate user preferences specified in terms of weights related to the interest dimensions *security*, *attractiveness*, and *crowdedness*.

On the basis of the requirements defined in gp and the item definitions in Table 2.8, a conjunctive query $\sigma_{[r_{11},r_{12},r_{22},r_{13},r_{14},r_{15}]}I$ results in: $\{t_1, t_3, t_5, t_6, t_7, t_9\}$, i.e., these items are consistent with the requirements defined in gp. Formula 2.5 can be used then to determine item-specific utilities on the basis of the group profile gp. For example, $utility(gp,t_1) = \sum_{d \in Dimensions} contribution(t_1,d) \times weight(gp,d) = contribution(t_1,security) \times weight(gp,security) + contribution(t_1,attractiveness) \times weight(gp,attractiveness) + contribution(t_1,crowdedness) \times weight(gp,crowdedness) = 5 \times 0.34 + 5 \times 0.48 + 2 \times 0.18 = 1.7 + 2.4 + .36 = 4.46$. The resulting utilities are shown in Table 2.17.

$$utility(gp,item) = \sum_{d \in Dimensions} contribution(item,d) \times weight(gp,d) \tag{2.5}$$

It can be the case that a set of user requirements is *inconsistent with all items* of an itemset. In such a situation, users of a constraint-based recommender have to adapt their requirements such that at least one solution can be identified. Related techniques will be discussed in the following section.

weights & requirements	u_1	u_2	u_3	u_4	u_5	gp
security	0.5	0.2	0.3	0.6	0.1	0.34 (AVG)
attractiveness	0.4	0.7	0.3	0.2	0.8	0.48 (AVG)
crowdedness	0.1	0.1	0.4	0.2	0.1	0.18 (AVG)
season	r_{11}: spring	r_{12}: spring	-	r_{14}: spring	-	r_{11}, r_{12}, r_{14}
topics	-	r_{22}: citytours	r_{13}: entertainment	-	r_{15}: citytours	r_{22}, r_{13}, r_{15}

Table 2.16.: Construction of a group profile (gp). User-specific weights regarding the interest dimensions *security*, *attractiveness*, and *crowdedness* are aggregated into gp using AVG. Furthermore, user requirements r_{ij} are combined into $R = \{r_{11}, r_{12}, r_{22}, r_{13}, r_{14}, r_{15}\}$.

2.7. Handling Inconsistencies

Since item retrieval in constraint-based recommendation is based on semantic queries (e.g., conjunctive queries), situations can occur where no solution can be identified for the given set of requirements (Felfernig et al., 2012), i.e., $\sigma_{[R]}I = \emptyset$ (R represents the union of requirements specified by individual group members and I represents the example itemset shown in Table 2.8). An example of such a situation is the following (adapted version of the examples introduced in the previous sections): $R = \{r_{11} : season = summer, r_{21} : eval = 5.0, r_{12} : season = summer, r_{13} : topics = entertainment, r_{14} : topics = entertainment, r_{15} : eval = 5.0\}$ where $\sigma_{[R]}I = \emptyset$. Also in the context of group recommendation scenarios, we are interested in how to change the requirements defined by group members in order to be able to come up with a recommendation consistent with the requirements of all group members.

In the *aggregated predictions* scenario, inconsistencies induced by requirements occur on the 'single user' level: a user specifies his/her requirements but no recommendation can be identified. In this context, diagnosis algorithms help to identify possible changes to the user requirements such that a recommendation can be identified. This way, it can be guaranteed that no user-specific inconsistent requirements are passed to the group level.

In the *aggregated models* scenario, the task of resolving inconsistent situations is a similar one: in the case of inconsistencies between requirements defined by a specific group member, diagnosis can actively support him/her in restoring consistency.[†††] However, even if the requirements of a user profile are consistent, integrating the requirements of individual users into a group profile gp can induce inconsistencies on the group level (Felfernig et al., 2016). In the aggregated models scenario, diagnosis also supports the achievement of *global consistency*, i.e., all joint preferences defined by individual group members allow the derivation of at least one solution. Table 2.18 shows the user requirements specified in our example.

The conflict sets induced by our example requirements (R) are: $CS_1 : \{r_{11}, r_{21}\}$, $CS_2 : \{r_{11}, r_{15}\}$, $CS_3 : \{r_{12}, r_{21}\}$, $CS_4 : \{r_{12}, r_{15}\}$, $CS_5 : \{r_{13}, r_{21}\}$, $CS_6 : \{r_{13}, r_{15}\}$, $CS_7 : \{r_{14}, r_{21}\}$, and $CS_8 : \{r_{14}, r_{15}\}$. If we resolve the conflicts by deleting the requirements r_{21} and r_{15}, a corresponding diagnosis (hitting set) $\Delta_1 =$

[†††] A discussion of algorithms for diagnosis determination can be found in (Reiter, 1987; Felfernig et al., 2004, 2009b, 2011b).

item	item contribution			utility(gp, t_i)
	secur.	*attr.*	*crowd.*	
t_1	5	5	2	4.46 $\surd$
t_2	4	4	4	4.0
t_3	3	5	1	3.6
t_4	4	3	5	3.7
t_5	3	4	1	3.12
t_6	3	3	1	2.64
t_7	2	3	3	2.66
t_8	4	4	4	4.0
t_9	3	5	1	3.6
t_{10}	3	3	3	3

Table 2.17.: Item utilities determined on the basis of the weights defined in gp (see Table 2.16). Only items t_i are taken into account that are consistent with the requirements in gp (others are shown greyed out). The $\surd$ symbol indicates the item with the highest utility.

$\{r_{21}, r_{15}\}$ can be identified. The second diagnosis is $\Delta_2 = \{r_{11}, r_{12}, r_{13}, r_{14}\}$. The determination of the diagnoses Δ_1 and Δ_2 is shown on the basis of the HSDAG approach (Hitting Set Directed Acyclic Graph) (Reiter, 1987) (see Figure 2.9). Table 2.18 includes a third diagnosis ($\Delta_3 = \{r_{11}, r_{21}, r_{15}\}$) which has been included to show that non-minimal diagnoses $\Delta_{\neg min}$ are not preferred by aggregation functions (see Tables 2.19 – 2.20). A corresponding subset ($\Delta \subset \Delta_{\neg min}$) exists that already fulfills the diagnosis properties. In our example, $\Delta_1 \subset \Delta_3$ holds, i.e., Δ_3 is a non-minimal diagnosis.

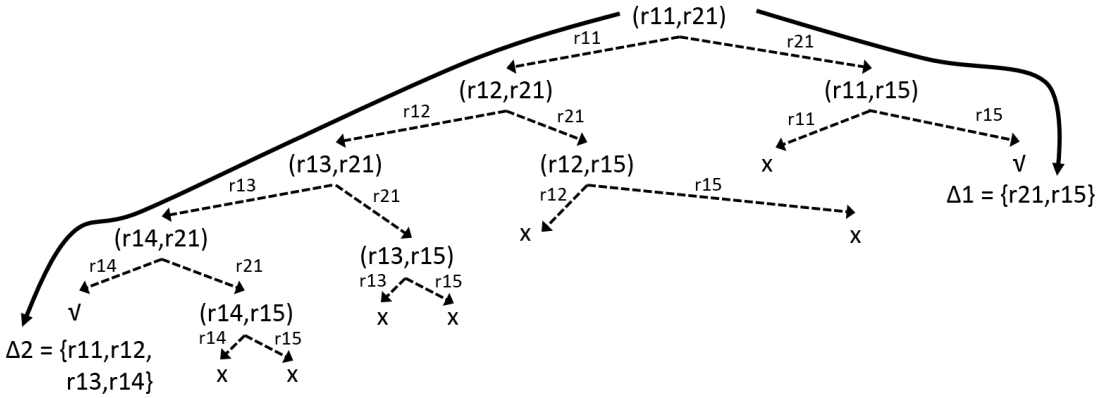

Figure 2.9.: Determination of the minimal diagnoses Δ_1 and Δ_2 using the HSDAG approach (Reiter, 1987) (paths to minimal diagnoses are denoted with $\surd$).

As different diagnosis candidates exist ($\Delta_1, \Delta_2, \Delta_3$), we have to figure out which one should be recommended to the group. Similar to the determination of recommendations, diagnosis candidates can be ranked on the basis of different aggregation functions. In Table 2.19 we sketch an approach to rank diagnoses depending on the number of requirements that have to be deleted/adapted by individual group members. Diagnosis Δ_1 has the *lowest number of needed changes* (*ADD*); consequently it can be rec-

requirement	Δ_i		
	Δ_1	Δ_2	Δ_3
$r_{11(season=0100)}$		•	•
$r_{21(eval=5.0)}$	•		•
$r_{12(season=0100)}$		•	
$r_{13(topic=entertainment)}$		•	
$r_{14(topic=entertainment)}$		•	
$r_{15(eval=5.0)}$	•		•

Table 2.18.: Example user requirements and related diagnoses in the *aggregated models* scenario (r_{ij} = requirement i of user j): $\Delta_1 = \{r_{21}, r_{15}\}$, $\Delta_2 = \{r_{11}, r_{12}, r_{13}, r_{14}\}$, and $\Delta_3 = \{r_{11}, r_{21}, r_{15}\}$. Δ_3 is a non-minimal diagnosis included to show that aggregation functions prefer minimal diagnoses.

ommended. *Least Misery* (LMS) recommends one out of $\{\Delta_1, \Delta_2\}$. As mentioned, we will not discuss diagnosis algorithms in this chapter; for a detailed discussion of diagnosis search and selection in group contexts we refer to (Felfernig et al., 2016).

diagnosis	changes per user					aggregation	
	u_1	u_2	u_3	u_4	u_5	*ADD*	*LMS*
Δ_1	1	0	0	0	1	2 $\surd$	1$\surd$
Δ_2	1	1	1	1	0	4	1$\surd$
Δ_3	2	0	0	0	1	3	2

Table 2.19.: Diagnosis recommendation in the *aggregated models* scenario based on (1) counting the needed changes per user and (2) *LMS*. The $\surd$ symbol indicates recommended diagnosis candidates.

Diagnosis ranking can be better personalized, if we assume that requirements have importance weights learned, for example, on the basis of previous group decisions (Felfernig et al., 2009b; Guo et al., 2017). Table 2.20 depicts an example of the determination of diagnosis utilities on the basis of weighted requirements – the utility of a diagnosis can be determined on the basis of Formula 2.6. That implements an additive aggregation strategy: the higher the sum of the individual weights $w(r_{ij})$, the higher the importance of the related requirements for the group members. Consequently, the lower the total importance of the included requirements, the higher the utility of the corresponding diagnosis (see Formula 2.6). In this setting, diagnosis Δ_2 outperforms Δ_1 (also Δ_3) since Δ_2 includes requirements less relevant for the individual group members. *Least Misery* (LMS) in this context analyzes (user-wise) attribute-specific estimated negative impacts of requirement deletions.

$$utility(\Delta) = \frac{1}{\sum_{r_{ij} \in \Delta} w(r_{ij})} \tag{2.6}$$

Remark. An issue for future work in this context is to analyze the possibility of combining the group profile (gp) with local user profiles. This could serve to assure consensus in the group earlier, and avoid efforts related to conflict resolution on the group level. If parts of the group profile are integrated into individual user profiles, this could also help to take into account the requirements of other group members at the very

Δ_i	weighted requirements						aggregation	
	$w(r_{11}) = 0.1$	$w(r_{21}) = 0.3$	w(r_{12}) = 0.1	w(r_{13}) = 0.1	q(r_{14})0.1	w(r_{15})=0.3	*utility*	*LMS*
Δ_1	0	0.3	0	0	0	0.3	1.67	0.3
Δ_2	0.1	0.0	0.1	0.1	0.1	0	2.5$\surd$	0.1$\surd$
Δ_3	0.1	0.3	0	0	0	0.3	1.42	0.3

Table 2.20.: Utility-based diagnosis recommendation in the *aggregated models* scenario. The $\surd$ symbol indicates the highest rated diagnosis.

beginning of the decision making process. Further details on how to determine personalized diagnoses on the basis of search heuristics can also be found in (Felfernig et al., 2009b, 2013b).

2.8. Critiquing-based Recommendation for Groups

Critiquing-based recommendation (Guzzi et al., 2011; Chen and Pu, 2012a) is based on the idea of showing *reference items* to users and allowing users to give feedback in terms of *critiques*. Critiques trigger a new critiquing cycle where *candidate items* (items that fulfill the critiques defined by the user*) are compared with regard to their utility as a new *reference item*. This utility is evaluated on the basis of (a) *similarity metrics* that estimate the *similarity* between a reference item and a candidate item and (b) the degree of *support* of the critiques already defined by a user.† Intuitively, the more similar a candidate item is with regard to the reference item and the more critiques it supports, the higher its utility. In the following, we assume that the determination of candidate items for a specific group member takes into account his/her previous critiques and the similarity between reference and candidate item. The utility of a candidate item as the next reference item can be determined on the basis of Formulae 2.7 – 2.9. In this context, $utility(c,r,u)$ denotes the utility of a candidate item c to act as a reference item for user u taking into account the current reference item r. Furthermore, $sim(c,r)$ determines the similarity between r and c. Finally, $support(c, critiques(u))$ evaluates the support candidate item c provides for the critiques defined by user u. In this context, *support* is measured in terms of (a) *consistency* between candidate item and critiques and (b) the *weight* of individual critiques (for example, older critiques could have a lower weight).

$$utility(c,r,u) = sim(c,r) \times support(c, critiques(u)) \tag{2.7}$$

$$support(c, critiques) = \Sigma_{crit \in critiques} consistent(c, crit) \times weight(crit) \tag{2.8}$$

$$consistent(c, crit) = \begin{cases} 1 & \text{if } \sigma_{[crit]}\{c\} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Let us assume that the first reference item (item r that is the first one shown to start a critiquing session) shown to each group member is t_1. Table 2.21 depicts example critiques defined thereafter on t_1 by the group members u_1, u_2, and u_3. We also assume that items used in the example correspond to the travel destinations itemset shown in Table 2.8. Finally, we assume *equal weights* for critiques.

*Different variants thereof exist in critiquing-based systems ranging from *taking into account only the most recent critique* to *all critiques in the critiquing history*.

†Also denoted as *compatibility score* (McCarthy et al., 2006).

user	1^{st} *critique*	2^{nd} *critique*
u_1	t_1:winter $\in$ season (cr_{11})	t_3:eval $>$ 3.3 (cr_{12})
u_2	t_1:nature $\in$ topics (cr_{21})	t_2:winter $\in$ season (cr_{22})
u_3	t_1:eval $>$ 4.5 (cr_{31})	t_4:citytours $\in$ topics (cr_{32})

Table 2.21.: A group-based critiquing scenario: each group member already specified two critiques (denoted as *critiquing history*). The reference item for the 1^{st} critiquing cycle is assumed to be $t_1(u_1,u_2,u_3)$, the reference items for the 2^{nd} critiquing cycle are $t_3(u_1)$, $t_2(u_2)$, and $t_4(u_3)$.

The similarities between potential combinations of reference items (r) and candidate items (c) are depicted in Table 2.22. The attributes *season* (EIB), *topics* (EIB), and *eval* (NIB) are taken into account.[‡] For example, sim(t_1,t_2) = s(t_1.*season.spring*, t_2.*season.spring*)$\times\frac{1}{9}$ + s(t_1.*season.summer*, t_2.*season.summer*)$\times\frac{1}{9}$ + s(t_1.*season.autumn*, t_2.*season.autumn*)$\times\frac{1}{9}$ + s(t_1. *season. winter*, t_2.*season.winter*)$\times\frac{1}{9}$ + s(t_1.*topics.citytours*, t_2.*topics.citytours*)$\times\frac{1}{9}$ + s(t_1. *topics.entertainment*, t_2.*topics.entertainment*)$\times\frac{1}{9}$ + s(t_1.*topics.nature*, t_2.*topics. nature*)$\times\frac{1}{9}$ + s(t_1.*topics.beach*, t_2.*topics.beach*)$\times\frac{1}{9}$ + s(t_1.*eval*, t_2.*eval*)$\times\frac{1}{9}$ = 0.66.

item	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
t_1	1.0	.66	.75	.32	.86	.88	.54	.61	.74	.77
t_2	-	1.0	.43	.64	.53	.54	.67	.96	.42	.64
t_3	-	-	1.0	.52	.88	.86	.54	.42	.99	.74
t_4	-	-	-	1.0	.4	.44	.53	.6	0.51	.56
t_5	-	-	-	-	1.0	.96	.42	.53	.89	.84
t_6	-	-	-	-	-	1.0	.43	.5	0.85	.88
t_7	-	-	-	-	-	-	1.0	.62	.53	.53
t_8	-	-	-	-	-	-	-	1.0	0.42	.6
t_9	-	-	-	-	-	-	-	-	1.0	.73
t_{10}	-	-	-	-	-	-	-	-	-	1.0

Table 2.22.: Items of Table 2.8 (similarity with regard to *season*, *topics*, and *eval*).

The selection of a new reference item in the critiquing scenario shown in Table 2.21 is depicted in Table 2.23. In this context, reference items are not considered potential candidate items, since the same item should not be presented in follow-up critiquing cycles. Each table entry represents the utility of a specific candidate item (from Table 2.8) with regard to a reference item. For example, $utility(c:t_2, r:t_3, u:u_1) = sim(t_2,t_3) \times support(t_2, critiques(u_1)) = 0.43 \times (0 \times 0.5 + 1 \times 0.5) = 0.21$ (two critiques, i.e., equal weights = 0.5).

If a user interacts with a critiquing-based recommender in *standalone* mode (critiques of other users are not taken into account), he/she receives recommendations related to his/her preferences (McCarthy et al., 2006). In parallel, critiques from individual group members can be forwarded to a group recommender. Different variants thereof are possible. For example, recommendations determined for a single user can

[‡]Similarity metrics (see (McSherry, 2004)) – we assume, *minval*=0 and *maxval*=5.

u	r	$utility(t_i, r, u)$									
		t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
u_1	t_3	-	.21	-	.52	0	.43	.54$\surd$	0	.49	.37
u_2	t_2	-	-	.21	.64	0	0	.67$\surd$	.48	.21	0
u_3	t_4	-	0	.26	-	.2	.44	.26	0	.25	.56$\surd$

Table 2.23.: Selection of new reference items based on the *utility* of candidate items t_i (calculation is based on Formula 2.7). We assume that previous reference items are not reference item candidates anymore (represented by '-' entries). The $\surd$ symbol denotes the selected new reference items.

also take into account the preferences of the whole group by simply taking into account some or all of the critiques stored in the group profile (McCarthy et al., 2006). In this context, weights regarding the trade-offs between the importance of user-individual critiques and critiques on the group level have to be specified.

Aggregated Predictions. The process of critiquing-based group recommendation using aggregated predictions is sketched in Figure 2.10.

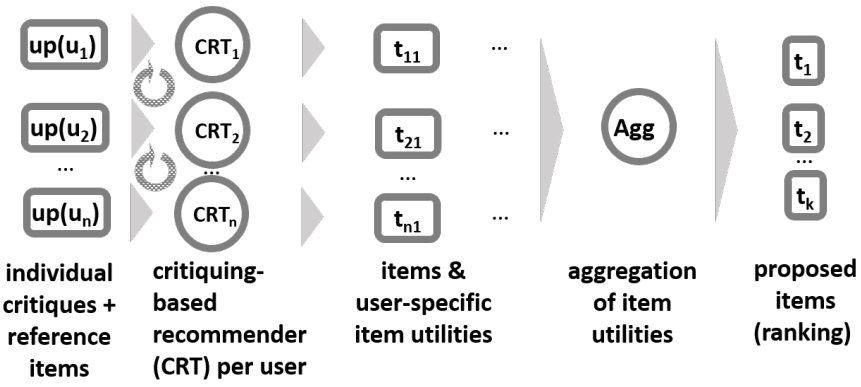

Figure 2.10.: Critiquing-based recommendation for groups with *aggregated predictions*. User preferences are constructed iteratively (conversational recommendation).

On the basis of an initial reference item, individual critiquing-based recommenders start the first critiquing cycle and – depending on user feedback – determine follow-up reference items. In other words, several interaction cycles preceed a decision. After individual group members have completed their selection process, the corresponding results (see, e.g., Table 2.21) can be used to determine a group recommendation. Table 2.24 depicts user-specific utilities of new items (the similarity values are taken from Table 2.23).
An alternative to the aggregation of item utilities (Table 2.24) is to *aggregate items* proposed by individual critiquing-based recommenders. A group recommendation can be determined, for example, by taking the item with the highest utility value per group member (Formula 2.7). The group recommendation is $\{t_7, t_{10}\}$. As discussed in (Guzzi et al., 2011), items can be proposed by group members and group members can provide counter-proposals that – with some likelihood – are acceptable to other group members.

c	utility(c,r,u) (score)			aggregation		
	$u_1(r:t_3)$	$u_2(r:t_2)$	$u_3(r:t_4)$	AVG	BRC	LMS
t_1	0 (1.5)	0 (2)	0 (1.5)	0	5	0
t_2	0.21 (4)	0 (2)	0 (1.5)	0.07	7.5	0
t_3	0 (1.5)	0.21 (5.5)	0.26 (6.5)	0.16	13.5	0
t_4	0.52 (8)	0.64 (8)	0 (1.5)	0.39	17.5	0
t_5	0 (1.5)	0 (2)	0.2 (4)	0.07	7.5	0
t_6	0.43 (6)	0 (2)	0.44 (8)	0.29	16	0
t_7	0.54 (9)	0.67 (9)	0.26 (6.5)	0.49 $\surd$	24.5 $\surd$	0.26 $\surd$
t_8	0 (1.5)	0.48 (7)	0 (1.5)	0.16	10	0
t_9	0.49 (7)	0.21 (5.5)	0.25 (5)	0.32	17.5	0.21
t_{10}	0.37 (5)	0 (2)	0.56 (9)	0.31	16	0

Table 2.24.: User-specific utilities of new items (see Formula 2.7). $\surd$ indicates the item with the best evaluation determined by the corresponding aggregation function.

Aggregated Models. Following this strategy, a group model (critiquing history on the group level) has to be generated (see Figure 2.11).

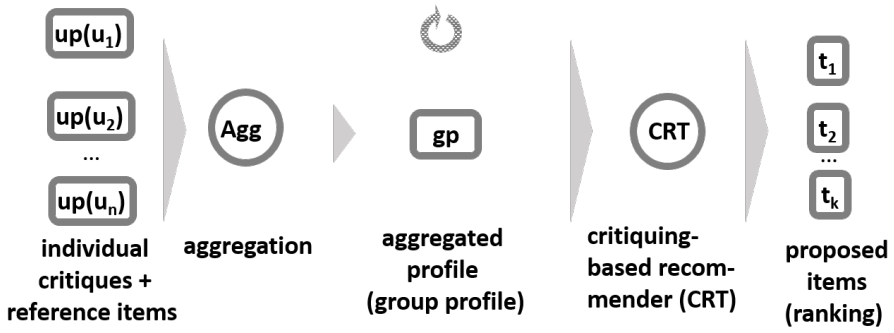

Figure 2.11.: Critiquing-based recommendation for groups with *aggregated models*. Group preferences are constructed iteratively (conversational recommendation).

On the basis of a group model (group profile - gp), a corresponding group recommendation can be determined. In order to build a group profile (gp), critiques defined by group members have to be aggregated. Table 2.25 depicts an example of the aggregation of group member specific critiquing histories into a group profile (gp). In this scenario, the aggregation of individual critiques can lead to a situation where none of the items completely fulfills the defined critiques (see the example group profile in Table 2.25). As a consequence, we have to identify recommendations which support as many critiques as possible. In order to determine a ranking for the different items, Formula 2.10 can be applied where $utility(t,gp)$ denotes the utility of item t with regard to the critiques part of the group profile gp, and *weight* represents the weight of a critique. In our example, we assume equal weights, however, weights can also be used to reduce the impact of less up-to-date critiques.

$$utility(t,gp) = \Sigma_{crit \in critiques(gp)} consistent(t,crit) \times weight(crit) \tag{2.10}$$

group G	group profile (defined by critiques) of G
$\{u_1, u_2, u_3\}$	winter $\in$ season, nature $\in$ topics, eval $>$ 4.5, citytours $\in$ topics

Table 2.25.: Set of critiques (=group profile gp) defined by the group $G = \{u_1, u_2, u_3\}$.

Table 2.26 represents a list of items (determined on the basis of Formula 2.10) and corresponding utilities with regard to the critiques contained in the group profile gp. For example, $utility(t_1, gp) = 0 \times 0.25 + 0 \times 0.25 + 0 \times 0.25 + 1 \times 0.25 = 0.25$.

item	utility
t_1	0.25
t_2	0.25
t_3	0.5
t_4	0.75 $\surd$
t_5	0.25
t_6	0.5
t_7	0.75 $\surd$
t_8	0.25
t_9	0.5
t_{10}	0.5

Table 2.26.: Group-specific utilities of new items determined on the basis of Formula 2.10. The $\surd$ symbol indicates items with the highest utility values.

2.9. Hybrid Recommendation for Groups

As already mentioned, hybrid recommendation helps to compensate specific limitations of one recommendation approach with the strengths of another one (Burke, 2002; DePessemier et al., 2017). We will now sketch hybridization in the context of group recommender systems (Berkovsky and Freyne, 2010; DePessemier et al., 2014, 2015).

Weighted. The idea of weighted hybrid recommendation is to combine the results received from individual recommenders into a corresponding group recommendation. Table 2.27 shows a simple example of applying weighted hybridization in the context of group recommendation. A collaborative recommender for groups (CF) based on the *aggregated models* (AM) strategy and a content-based filtering recommender (CBF) for groups based on the *aggregated predictions* (AP) strategy return the item rankings shown in Table 2.27. The *Borda Count* (BRC) strategy (see Table 2.2) can now be applied to aggregate the corresponding scores.

Mixed. Hybrid recommendation based on the mixed strategy combines the recommended items returned by the individual recommenders (see Table 2.28).

item	recommender-specific evaluations (scores)		aggregation
	CF ratings (AM,AVG)	CBF similarities (AP,LMS)	BRC
t_1	4.9 (8)	0.81 (9)	17 $\surd$
t_2	2.2 (1)	0.32 (1)	2
t_3	5.0 (9)	0.66 (7)	16
t_4	4.3 (7)	0.61 (6)	13
t_5	1.5 (0)	0.2 (0)	0
t_6	3.8 (3)	0.55 (5)	8
t_7	3.4 (2)	0.49 (4)	6
t_8	4.1 (4)	0.45 (3)	7
t_9	4.2 (5.5)	0.33 (2)	7.5
t_{10}	4.2 (5.5)	0.79 (8)	13.5

Table 2.27.: Recommendation results of two group recommenders (CF based on *aggregated models* (AM) and CBF based on *aggregated predictions* (AP)) as list of ranked items are aggregated on the basis of *Borda Count* (BRC). The $\surd$ symbol indicates the item with the best evaluation.

In our example, the rankings returned by two group recommenders are aggregated using the fairness (FAI) function where items are included in the final recommendation following the zipper principle, i.e., the item ranked highest by the CBF recommender is integrated first, then the item ranked highest by the CF recommender is integrated into the recommendation result, and so on.

item	recommender-specific rankings		aggregation
	CF (AM,AVG)	CBF (AP,LMS)	FAI (ranking)
t_1	10	9	10
t_2	2	1	1 $\surd$
t_3	7	6	7
t_4	6	5	6
t_5	1	4	2
t_6	3	7	4
t_7	5	3	5
t_8	9	8	9
t_9	4	2	3
t_{10}	8	10	8

Table 2.28.: Recommendation results of two group recommenders (CF and CBF) as a list of ranked items aggregated on the basis of *Fairness* (FAI) that implements the zipper principle (alternate inclusion of best ranked items – the item ranked highest by CBF is integrated first). $\surd$ indicates the item with the highest ranking.

2.10. Matrix Factorization for Groups

Up to now, we have discussed ways to apply the recommendation approaches of collaborative filtering, content-based filtering, constraint-based, critiquing-based, and hybrid recommendation in group contexts. *Matrix factorization* is a popular approach to collaborative filtering based recommendations (Koren et al., 2009) . The underlying idea is to explain ratings by characterizing items and users on the basis of a set of factors. The original user $\times$ item matrix is separated into two lower-dimensional ones that explain user item interactions on the basis of the mentioned factors (see Table 2.29).

In this context, each item t is associated with a vector q_t that describes to which extent t represents the factors. Furthermore, each user u is associated with a vector p_u that describes to which extent the factors are important for the user. Finally $\hat{r}_{ui} = q_i^T p_u$ represents an approximation of a user's u rating of t (r_{ui} denotes a user's real rating). More formally, we factorize the rating matrix $\boldsymbol{R} \in \mathbb{R}^{n \times m}$ containing known ratings for n users and m items into matrices $\boldsymbol{P} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{Q} \in \mathbb{R}^{m \times k}$ such that $\boldsymbol{P}\boldsymbol{Q}^T$ closely approximates $\boldsymbol{R}$. In literature and practice, there are several possibilities to measure and minimize the approximation error of the factorization. A popular choice for the approximation error is the sum of the squared errors combined with a simple regularization term, e.g. $\sum_{r_{u,i} \neq \bullet} (r_{u,i} - \mu - \vec{p}_u^T \vec{q}_i)^2 + \lambda(||\vec{p}_u||^2 + ||\vec{q}_i||^2)$, where μ is the global rating average and $\bullet$ represents an unknown rating. Minimization of the error is typically computed with a variant of the gradient descent method.

An approach to the application of matrix factorization in the context of group recommendation scenarios is presented in (Ortega et al., 2016). The authors introduce two basic strategies denoted as *After Factorization* (AF) and *Before Factorization* (BF). When using AF (see Table 2.30), user-individual matrix factorization is performed in order to identify user-specific factors which are thereafter aggregated (e.g., by determining the average (AVG) of the user-individual factor values). When using BF (see Table 2.31), first user-individual item ratings are aggregated into a group profile, followed by a matrix factorization approach. These two basic variants follow the idea of *aggregated predictions* (AF) and *aggregated models* (BF).

	i1	i2	i3	i4	i5	i6	i7	i8
u1	5	1		2	2			2
u2	1	4	2		5	1		4
u3			3	5		1	2	4
u4	4		5	3		5	3	
u5	4	1		1		4	3	
u6	4		1	1		5		1
u7		2	2		2		4	1
u8	4		3			4	3	

(a) Rating matrix $\boldsymbol{R}$

	$p_{u,1}$	$p_{u,2}$	$p_{u,3}$
u1	0.27984223	1.33194126	-0.25748666
u2	-0.13639896	-1.00299326	1.09421098
u3	0.29145967	-1.08249042	0.85824434
u4	1.29398513	0.94631031	0.77574863
u5	0.25258519	0.72222304	-1.20079938
u6	-1.26795635	1.16829146	-0.11806872
u7	-0.82074846	0.92881098	-0.20635514
u8	0.15691092	0.64969789	0.53725284

(b) User factors $\boldsymbol{P}$

	$q_{i,1}$	$q_{i,2}$	$q_{i,3}$
i1	0.23698436	1.38151089	-0.23953783
i2	-0.19159424	-1.01718827	0.59249957
i3	1.60559024	0.21567611	-0.26351522
i4	0.3814163	-0.94038814	0.9485212
i5	0.43533279	-0.3900103	1.52692825
i6	0.1258419	1.88675619	0.13922563
i7	-0.47012841	0.65365324	0.03900384
i8	0.98047664	-0.63261944	0.51537004

(c) Item factors $\boldsymbol{Q}$

Table 2.29.: We factorize the rating matrix $\boldsymbol{R}$ containing known ratings for $n = 8$ users and $m = 8$ items into matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ such that $\boldsymbol{P}\boldsymbol{Q}^T$ closely approximates $\boldsymbol{R}$. For illustration purposes, we minimize the sum of the squared errors of the approximation together with a simple squared L2-norm regularization term. We set $k = 3$ factors and regularization parameter to $\lambda = 0.02$. We initialize $\boldsymbol{P}$ and $\boldsymbol{Q}$ randomly and optimize with the gradient descent algorithm. Note that factorization brings similar users close to each other in the factor space (c.f. factors of users $u2$ and $u3$ in $\boldsymbol{P}$), whereas dissimilar users are projected further apart (c.f. factors of users $u1$ and $u2$ in $\boldsymbol{P}$).

	$p_{G,1}$	$p_{G,2}$	$p_{G,3}$
G	0.144968	-0.25118	0.56499

(a) AF: Group factors

	i1	i2	i3	i4	i5	i6	i7	i8
G	2.40	3.41	2.88	3.68	3.87	2.47	2.64	3.44

(b) AF: Predicted ratings

Table 2.30.: In the *After Factorization* (AF) approach the group of users is factorized by merging factors of users (e.g., by calculating averages) in a given group. In our example, we group three users from $G = \{u1, u2, u3\}$. Note that users $u2$ and $u3$ are highly similar to each other but are highly dissimilar to user $u1$. Thus, we expect the group ratings to be biased towards the ratings of users $u2$ and $u3$ as group ratings for items $i1$ (lower because of a low rating from user $u2$) and $i2$ (higher because of a high rating of user $u2$) show.

Due to its simplicity, AF is efficiently calculated and provides a solid baseline for group recommendation approaches based on matrix factorization. However, in practice BF gives significantly better prediction results on larger datasets and for larger groups. For more details on matrix factorization based recommendation approaches we refer to (Koren et al., 2009). Approaches to apply matrix factorization in the context of group recommendation scenarios are discussed in (Hu et al., 2011; Ortega et al., 2016).

	$p_{G,1}$	$p_{G,2}$	$p_{G,3}$
G	0.12873	-0.56466	-0.03111

(a) BF: Group factors

	i1	i2	i3	i4	i5	i6	i7	i8
G	2.11	3.38	2.94	3.40	3.08	1.80	2.42	3.32

(b) BF: Predicted ratings

Table 2.31.: In the *Before Factorization* (BF) approach a virtual group user is created from the rating matrix by e.g. calculating the average ratings (AVG) for the users from a given group. In the next step, the group factors are calculated from the given factorization by calculating the (Ridge) regression coefficients on the ratings of the virtual user. Finally, the group factors allow us to predict group ratings. The intuition behind BF approach is that the virtual user is a better representation of the users group than a simple aggregation of users factors. In our example, BF predicts a significantly lower rating than AF for item $i6$ because there is much stronger evidence in the data for a low rating (two 1-star ratings).

2.11. Conclusions and Research Issues

In this chapter, we have introduced different group recommendation techniques which are based on the recommendation approaches for individual users. We showed how related group recommendation scenarios can be designed for collaborative filtering, content-based filtering, constraint-based including utility-based recommendation, critiquing-based, and hybrid recommendation. In this context, we focused on a discussion of the two aggregation strategies: (1) *aggregated predictions* (*items*) and (2) *aggregated models*. In (1), recommendations are determined for individual group members and then aggregated. In (2), the preferences of group members are aggregated, and recommendations are then determined on the basis of information contained in the integrated group profile. An issue already solved in a couple of *person-2-person* recommendation environments is which algorithms can be used to find a person that fits another person with regard to a set of predefined criteria. An online dating application is reported, for example, in (Wobcke et al., 2015). Another application is the identification of experts to support the answering of specific questions (McDonald and Ackerman, 2000). A related issue, especially relevant in the context of group decision making, is *group synthesis*, i.e., the identification of a group that is able to solve a specific problem or to make a decision. Initial work on group synthesis in the context of open innovation scenarios can be found in (Brocco and Groh, 2009; Hong et al., 2014). A major criteria is to identify a group that is able to solve a given (decision) task, taking into account availability aspects such as engagement in other projects. This scenario can become even more complex if we want to configure a set of groups to solve a specific task. Consider the following university-based task: Given that there are 300 students registered in a software engineering course, divide the population into groups of 6, such that each group is best suited to complete a specific project. A related issue is the analysis of inter-group influences, for example, in which way *influential groups* influence *susceptible groups* (Recalde, 2017). Further research issues are related to the topics of *evaluating group recommenders*, *explaining group recommendations*, *taking into account group dynamics*, and *counteracting biases* that trigger suboptimal decisions.

Chapter 3

An Analysis of Group Recommendation Heuristics for High- and Low-Involvement Items

Parts of the contents of this chapter have been published in (Felfernig et al., 2017a). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, data analysis, and evaluation.

3.1. Abstract

Group recommender systems are based on aggregation heuristics that help to determine a recommendation for a group. These heuristics aggregate the preferences of individual users in order to reflect the preferences of the whole group. There exist a couple of different aggregation heuristics (e.g., most pleasure, least misery, and average voting) that are applied in group recommendation scenarios. However, to some extent it is still unclear which heuristics should be applied in which context. In this chapter, we analyze the impact of the item domain (low-involvement vs. high-involvement) on the appropriateness of aggregation heuristics (we use *restaurants* as an example of low-involvement items and *shared apartments* as an example of high-involvement ones). The results of our study show that aggregation heuristics in group recommendation should be tailored to the underlying item domain.

3.2. Introduction

In contrast to single user recommender systems (Jannach et al., 2010; Jameson et al., 2015), group recommenders focus on the recommendation of items to groups (Jameson and Smyth, 2007; Masthoff, 2011). For example, Masthoff (Masthoff, 2004) presents concepts for television item sequencing for groups, O'Connor et al. (O'Connor et al., 2001) present a collaborative filtering based approach to movie recommendation for groups, McCarthy et al. (McCarthy et al., 2006) present a critiquing-based recommendation approach for groups of users (skiing holiday package selection), Ninaus et al. (Ninaus et al., 2014a) demonstrate the application of group recommendation techniques in software requirements engineering, Jameson (Jameson, 2004) presents user interface concepts that help to elicit and aggregate

user preferences in the tourism domain (and beyond), and Stettinger et al. (Stettinger et al., 2015a,b) introduce a domain-independent recommendation-enhanced decision support environment for groups named CHOICLA.

There exist different approaches to determine recommendations for groups (Jameson and Smyth, 2007). In most scenarios, preferences of individual group members are aggregated on the basis of aggregation functions (heuristics) (Masthoff, 2011). The outcome of such an aggregation reflects the preferences of the whole group with regard to a given set of items. In the study presented in this chapter, we simulate a situation where users have explicitly specified their preferences with regard to a set of items (restaurants and shared apartments) and aggregation heuristics are then used to infer the corresponding group preferences. In this context, the task of each study participant was to analyze a given set of user preferences with regard to an item set and then to provide a recommendation for the whole group.

Group recommender systems can be regarded as tools that support groups in decision making processes. Depending on the type of item, users tend to invest more or less time until a final decision is taken. For example, a car purchase comes along with a long-term decision process where different alternatives are compared in-depth against each other. In contrast, when choosing a restaurant, the corresponding decision is typically taken rather fast. An important question to be answered in this context is whether the underlying decision heuristics differ since this has a major impact on the development of group recommender systems.

Depending on the decision scenario, humans tend to achieve an acceptable trade-off between effort and accuracy related to a decision making process and the corresponding outcome (Payne et al., 1993). *Satisficing* (Simon, 1955) is a related term that describes a human decision behavior where the first alternative is chosen that satisfies the wishes and needs of a user. Finally, items with high related decision efforts are often denoted as *high-involvement items* whereas items with low related decision efforts are denoted as *low-involvement items* (Petty et al., 1983). The impact of suboptimal decisions regarding high-involvement items is much higher compared to low-involvement items. For instance, a suboptimal decision in the shared apartment domain manifests, e.g., in search efforts for a new apartment, unnecessary payments for the old apartment, relocation costs, and additional time efforts. In the restaurant domain, the effects of a suboptimal decision are typically negligible.

To the best of our knowledge, in-depth analyses of the selection of preference aggregation heuristics depending on the item domain do not exist. Related work is presented in (Masthoff, 2011) where individual aggregation heuristics are compared in the movie domain without further comparing the heuristics in other item domains. We present the results of a study that investigates the impact of item type (high-involvement vs. low-involvement) on the chosen decision strategy.*

The remainder of this chapter is organized as follows. In Section 3.3 we shortly introduce the aggregation heuristics that are often used in group recommendation scenarios. In Section 3.4 we analyze to which extent the item type has an impact on chosen decision heuristics and which heuristics have the highest prediction quality in specific item domains. With the final section we discuss research issues and conclude the chapter.

*The work presented in this chapter has been partially conducted within the scope of the research projects WeWant (basic research project funded by the Austrian Research Promotion Agency) and OpenReq (Horizon 2020 project funded by the European Union).

3.3. Group Recommendation Heuristics

The scenario in our study is based on the assumption that each (hypothetical) member of a group explicitly specifies her preferences w.r.t. a given set of items. On the basis of this preference specification, corresponding aggregation functions (heuristics) aggregate preferences to a group model that represents the inferred preferences of the whole group. An example of a setting where each group member has already specified his/her preferences is depicted in Table 3.1.

	restaurant 1	restaurant 2	restaurant 3
1^{st} user	5	1	3
2^{nd} user	4	2	3
3^{rd} user	5	1	3
4^{th} user	4	2	3
AVG	**5**	2	3
LMIS	**4**	1	3
MPLS	**5**	2	3
MGD	**4 or 5**	1 or 2	3
ENS	**5**	2	3
MUL	**400**	4	81

Table 3.1.: Example setting: four group members evaluated restaurants. Study participants had to recommend one "winner" item per setting. The individual decision heuristics *AVG (average)*, *LMIS (least misery)*, *MPLS (most pleasure)*, *MGD (minimal group distance)*, *ENS (ensemble voting)*, and *MUL (multiplicative)* will recommend "restaurant 1" to the group.

The following set of aggregation functions was used within the scope of our study (corresponding examples of the application of these functions are depicted in Table 3.1).

Average (Formula 3.1) returns the average (in our example rounded to the nearest whole number) voting for item s as recommendation for the whole group. For example, the AVG value for *restaurant 1* is 5.

$$AVG(s) = \frac{\Sigma_{u \in Users} eval(u,s)}{|Users|} \tag{3.1}$$

Least Misery (Formula 3.2) returns the lowest voting for item s as group recommendation. For example, the LMIS value for *restaurant 1* is 4.

$$LMIS(s) = min(\bigcup_{u \in Users} eval(u,s)) \tag{3.2}$$

Most Pleasure (Formula 3.3) returns the highest voting for item s as group recommendation. For example, the MPLS value for *restaurant 1* is 5.

$$MPLS(s) = max(\bigcup_{u \in Users} eval(u,s)) \tag{3.3}$$

Minimal group distance (Formula 3.4) returns a rating d which has the minimum distance to the ratings of group members. For example, the MGD value for *restaurant 1* is 4 or 5.

$$MGD(s) = arg \min_{d \in \{1..5\}} (\Sigma_{u \in Users} |eval(u,s) - d|) \tag{3.4}$$

Ensemble voting (Formula 3.5) returns the majority value of the individual decision strategies. The majority of the following individual voting strategy results would be used to calculate the ensemble voting value: H = {AVG, LMIS, MPLS, MGD}. For example, the ENS value for *restaurant 1* is 5 because the value 5 occurs 3 times (result of AVG, MPLS, MGD) and the value 4 occurs only 2 times (result of LMIS, MGD).

$$ENS(s) = maxarg_{(d \in \{1..5\})} (\#(\bigcup_{u \in Users} eval(u,s) = d)) \tag{3.5}$$

Multiplicative heuristic (Formula 3.6) multiplies the rating values of all users for item s. For example, the MUL value for *restaurant 1* is 400 (5*4*5*4 = 400).

$$MUL(s) = \prod_{u \in Users} eval(u,s) \tag{3.6}$$

In the example depicted in Table 3.1, each of the used heuristics would recommend *restaurant 1* which is the dominating alternative in this scenario.

3.4. User Study

Overview. The overall goal of the user study was to figure out in which way decision heuristics of users change depending on the corresponding item domain. As an example of a high-involvement item we chose *shared apartments*, as an example of low-involvement items we chose *restaurants*.

We conducted a user study with students from two Austrian universities.[†] In the data collection phase, N=420 subjects participated in the study where each participant had to perform two tasks: (1) *select* a restaurant for the given preferences of a synthesized group (each group had four group members – an example of such a synthesized setting is depicted in Table 3.1) with regard to a set of three restaurants and (2) *explain* the selection (recommendation). Overall, each of 20 different tasks (see Table 3.3) received about 20 evaluations from different study participants. The overall idea was that study participants were confronted with synthesized group settings (see, e.g., Table 3.1) and had to provide feedback on which item they would recommend to the group. This approach to analyze the decision making of groups is referred to as user as wizard evaluation method (Masthoff, 2006).

Evaluations of group members were simulated on the basis of six evaluation patterns (see Table 3.2) which follow a symmetric distribution of user preferences as follows.

- *average support (AV)*: there is an average support by each group member and none of the group members has a strong preference regarding acceptance or rejection.
- *disagreement (DIS)*: there is no clear opinion about the item, i.e., evaluations range from positive to negative.

[†]Graz University of Technology (www.tugraz.at) and Alpen-Adria Universität Klagenfurt (www.aau.at).

- *majority positive (MAP)*: the majority has evaluated the item positively, only one user does not support the item.
- *majority negative (MAN)*: the majority has evaluated the item negatively, only one user likes the item.
- *no support (NO)*: none of the group members prefers the item, i.e., the group as a whole refuses the item.
- *full support (FULL)*: the item is a preferred one for each group member, i.e., fully supported by all group members.

Table 3.2 depicts the six different patterns used within the scope of our study. For example, in the task depicted in Table 3.1, pattern *FULL* was used for *restaurant 1*, pattern *NO* was used for *restaurant 2*, and pattern *AV* was used for *restaurant 3*.

pattern	1*st* user	2*nd* user	3*rd* user	4*th* user
1 (AV)	3	3	3	3
2 (DIS)	1	2	3	4
3 (MAP)	1	4	5	4
4 (MAN)	5	2	1	2
5 (NO)	1	2	1	2
6 (FULL)	5	4	5	4

Table 3.2.: Patterns of user preferences (evaluations) used in the study, for example, pattern 6 (FULL) reflects a situation where all group members evaluate the alternative very positively.

The complete set of tasks that were generated on the basis of the patterns depicted in Table 3.2 is depicted in Table 3.3. This table is the result of generating all possible combinations of three different patterns out from 6 patterns. The sequence in which the three patterns have been displayed to study participants was randomized. Each study participant had to solve two tasks, i.e., to select two items in two different settings (one in a restaurant and the other one in a shared apartment group decision context).

On the basis of the collected dataset, we evaluated the prediction quality of the group decision heuristics *AVG, LMIS, MPLS, MGD, ENS*, and *MUL* where *precision* (Herlocker et al., 2004) was measured in terms of the ratio between the number of correctly predicted group decisions (recommendations of study participants were interpreted as corresponding group decisions) and the overall number of predictions. Our hypothesis ($H1$) in this context was that for *the same overall combinations of patterns, study participants apply different decision heuristics depending on the item domain*, i.e., depending on the item domain (and related basic involvement type), different decision strategies are applied.

Within the scope of the study, participants had to provide *explanations* as to why they recommended (selected) a specific item for a defined group setting. This information is the basis for our second hypothesis ($H2$): *depending on the item type, different types of explanations are used for the selected item.* The explanation type was determined on the basis of sentiments which have been manually extracted from textual explanations provided by the study participants.

task	1^{st} pattern	2^{nd} pattern	3^{rd} pattern	dominance
1	1	2	3	n
2	1	2	4	n
3	1	2	5	n
4	1	2	6	y
5	1	3	4	n
6	1	3	5	n
7	1	3	6	y
8	1	4	5	n
9	1	4	6	y
10	1	5	6	y
11	2	3	4	n
12	2	3	5	y
13	2	3	6	y
14	2	4	5	n
15	2	4	6	y
16	2	5	6	y
17	3	4	5	n
18	3	4	6	y
19	3	5	6	y
20	4	5	6	y

Table 3.3.: Tasks used in the user study. N=20 tasks represent all possible combinations of three out of six patterns (see Table 3.2). *Dominance* denotes the fact that the item set used in the task includes a *dominant item*, i.e., an item that is not outperformed by another item in terms of a user evaluation. For example, *restaurant 1* in Table 3.1 is a dominating item.

Results. Our goal was to figure out whether study participants used different decision strategies depending on the item domain (restaurants and shared apartments). First, we analyzed tasks in the dataset which included a *dominating item*, i.e., *an item that outperforms alternative items in at least one user evaluation and is not outperformed by other items*. For example, *restaurant 1* in Table 3.1 outperforms the other restaurants in all corresponding user evaluations.

Table 3.4 summarizes the results of this analysis for the tasks which include dominating items: in both item domains (*restaurants* and *shared apartments*), the *MPLS* heuristics had the highest precision compared to *AVG, LMIS, MGD, ENS* and *MUL* (chi-square test, $p < 0.05$). In all of our evaluations, the rather high precision rates can be explained by the low number of alternatives used in the individual task settings (the study participants had to compare only three items).

Second, we analyzed task settings which did not include dominating items. Table 3.5 summarizes the corresponding results: in the restaurant domain, the *AVG* heuristic had the highest precision ($p < 0.05$). In the shared apartment domain, *LMIS* significantly outperformed the other heuristics ($p < 0.05$). Consequently, in settings with no clear winner (settings where no dominating alternative is included), the choice of a decision heuristic depends on the item domain.

domain	AVG	LMIS	MPLS	MGD	ENS	MUL
restaurants	90.7%	91.2%	**94.3%**	90.7%	92.0%	90.7%
apartments	92.9%	93.4%	**93.7%**	92.9%	93.6%	92.9%

Table 3.4.: Precision of decision heuristics in the domains of *restaurants* and *shared apartments* for tasks only *including* dominating items.

domain	AVG	LMIS	MPLS	MGD	ENS	MUL
restaurants	**81.2%**	75.0%	48.3%	79.1%	37.2%	70.9%
apartments	74.1%	**83.3%**	35.7%	73.0%	22.1%	69.0%

Table 3.5.: Precision of decision heuristics in the domains of *restaurants* and *shared apartments* for tasks *not including* dominating items.

Table 3.6 summarizes the precision of decision heuristics including dominating and non-dominating items in the domains restaurant and shared apartments. In restaurant domain, the AVG heuristic outperforms best and in the shared apartment domain, the LMIS heuristic significantly outperforms all the other heuristics.

domain	AVG	LMIS	MPLS	MGD	ENS	MUL
restaurants	**86.5%**	83.9%	73.6%	85.5%	67.4%	81.8%
apartments	84.5%	**88.9%**	67.6%	84.0	61.5%	85.7%

Table 3.6.: Precision of decision heuristics in the domains of *restaurants* and *shared apartments* for tasks including *dominating* and *non-dominating* items.

In order to understand the reasons why study participants selected certain alternatives, we applied a manual sentiment analysis to analyze the *explanations* provided by the study participants (see Table 3.7). All explanations were analyzed with regard to the dimensions *dominance*, *fairness*, and *consensus*. In both item domains (restaurants and shared apartments), item *dominance* was the preferred way of explaining item recommendations. This can be explained by the fact that group recommendations identified by a study participant are in many cases considered as the best ones for the group (i.e., are considered as dominating the alternative ones).
If the task setting did not include dominating items, the share of consensus- and fairness-related explanations increased. In the restaurant domain, explanations more referred to the aspect of *consensus* whereas in the shared apartment domain, the aspect of *fairness* plays a more important role. Typically, restaurants are related to low-involvement decisions where the misery of a minority seems to be more acceptable compared to high-involvement decisions. The results of our sentiment analysis are summarized in Table 3.7.

In tasks that included dominating items, the explanations of users predominantly referred to the *dominance* of a specific item, for example, *I recommend item x since it dominates the other alternatives and thus is clearly the best recommendation in the given setting*. In situations with no clear winner, i.e., no dominating items were included in the task setting, the share of explanations related to the dimensions

dimensions	dom:restaurants	dom:apartments	non-dom:restaurants	non-dom:apartments
dominance	93.1%	93.3%	62.6%	54.1%
fairness	2.3%	3.3%	13.9%	23.5%
consensus	4.6%	3.3%	23.5%	22.4%

Table 3.7.: Explanation focus depending on the item domain. The sentiment dimensions used in our analysis were *dominance* of an item, *fairness* with regard to every group member, and *consensus* within the group. Dominating alternatives in the item set (columns dom:restaurants and dom:apartments) trigger more explanations regarding item dominance. In scenarios that do not include dominating alternatives, other dimensions play a more important role. In apartment decisions, fairness plays a more important role. Finally, consensus plays a role in both, apartment and restaurant decisions.

of *consensus* and *fairness* increases. From the three analyzed aggregation heuristics, *AVG* reflects the idea of *consensus*, i.e., items are recommended that represent a kind of trade-off for all group members. In this context, for example, completely negative evaluations of a single user do not necessarily prevent the recommendation of the corresponding item. *Fairness* is reflected by the *LMIS* heuristic since low evaluations of even single users have a much stronger impact on the recommendation and in most of the cases avoid the recommendation of items that are not preferred by a minority of group members.

3.5. Conclusions and Future Work

In this chapter we have presented the results of a user study that focused on the analysis of the existing differences in used decision strategies depending on the item type. For the two domains of restaurants and shared apartments, we could show the existence of different aggregation heuristics especially in situations where there is no clearly dominating item. We consider our work as a first step towards a more in-depth analysis of the usage of aggregation heuristics in different item domains. Within the scope of our future work we will focus on the analysis of further group decision heuristics and compare their item domain-specific sensitivity. For example, we will analyze the impact of integrating different aspects of risk-awareness into the design of group recommendation heuristics. In this context we will also analyze variations in the number of items and group members. Furthermore, we will analyze the impact of factors such as gender, cultural background, and histories of decisions already taken by the same group on the decision making approach applied by the group. Finally, we will focus on the analysis of different possible ways to visualize and explain the current status of a decision process with the overall objective to increase consensus in the group and make high-quality decisions more efficiently.

Chapter 4

Beyond Item Recommendation: Using Recommendations to Stimulate Knowledge Sharing in Group Decisions

Parts of the contents of this chapter have been published in (Atas et al., 2017). The author of this thesis provided major parts of this chapter in terms of writing and literature research.

4.1. Abstract

The intensity of domain knowledge exchange among group members is an important factor that directly influences group decision quality. The more frequent information is exchanged among group members, the higher the quality of the corresponding decision. In this chapter we present results of an empirical study conducted with groups of students – the task of each group was to take a decision regarding the exam topics the group prefers. This group decision had to be taken on the basis of a group decision support environment with included recommendation functionality and a discussion forum that allows for information exchange among group members. Depending on the included variant of the group recommendation algorithm, groups received recommendations that varied in terms of recommendation diversity. The results of the study show that increased recommendation diversity leads to an increased degree of information exchange among group members.

4.2. Introduction

Single user recommender systems focus on the recommendation of items to individuals (Jannach et al., 2010; Ricci et al., 2011). In contrast, group recommender systems* determine item recommendations that fit the preferences of group members (Jameson and Smyth, 2007; Masthoff, 2011). Table 4.1 provides an overview of example group recommendation environments. Jameson (Jameson, 2004) introduces a prototype application that supports groups of users to elicit and aggregate user preferences with regard to holiday destinations. Masthoff (Masthoff, 2004) introduces concepts for television item sequencing for groups of

*The work presented in this chapter has been conducted within the scope of the Horizon 2020 project OpenReq (Intelligent Recommendation & Decision Technologies for Community-Driven Requirements Engineering).

users on the basis of different models from social choice theory (see also (Masthoff, 2011)). O'Connor et al. (O'Connor et al., 2001) present a collaborative filtering based approach to movie recommendation that determines recommendations for groups of users. Ninaus et al. (Ninaus et al., 2014a) demonstrate the application of group recommendation technologies in software requirements engineering scenarios where stakeholders are in charge of cooperatively developing, evaluating, and prioritizing requirements. Kudenko et al. (Kudenko et al., 2003) propose a system which helps a group of users while purchasing a product from an electronic catalog and mediates a group discussion with the goal to achieve consensus. McCarthy et al. (McCarthy et al., 2006) introduce a critiquing-based recommendation approach that supports groups of users in a skiing holiday package selection process. Finally, CHOICLA (Stettinger et al., 2015b) is a group decision support environment which includes group recommendation technologies in a domain-independent fashion – related example application domains are personnel decisions and restaurant selection. For a detailed overview of existing group recommender applications we refer to Jameson and Smyth (Jameson and Smyth, 2007) and Boratto et al. (Boratto and Carta, 2015).

system	domain	reference
TRAVEL DECISION FORUM	tourist destinations	(Jameson, 2004)
POLYLENS	movies	(O'Connor et al., 2001)
INTELLIREQ	software requirements	(Ninaus et al., 2014a)
CATS	ski holiday packages	(McCarthy et al., 2006)
CHOICLA	domain-independent, e.g., personnel decisions	(Stettinger et al., 2015b)

Table 4.1.: Example group recommender systems. For an in-depth discussion of group recommender applications we refer to (Jameson and Smyth, 2007; Boratto and Carta, 2015).

Also in the context of recommender systems, decision biases frequently occur and can lead to low-quality decisions (Chen et al., 2013; Felfernig, 2014; Jameson et al., 2015). Masthoff and Gatt (Masthoff and Gatt, 2006) report possible approaches for the prediction of group member satisfaction with recommendations – in this context, *conformity* and *emotional contagion* are stated as major influence factors. Felfernig et al. (Felfernig et al., 2011b) and Stettinger et al. (Stettinger et al., 2015a) discuss the impact of *conformity* on group decision making and report an increasing diversity of the preferences of group members the later individual preferences are disclosed. Chen and Pu (Chen and Pu, 2012b) show how emotional feedback of group members can be integrated in a music recommendation system. An outcome of their study is that emotional feedback can help to enhance the mutual awareness regarding the preferences of other group members.

Knowledge exchange between group members can have a major impact on decision quality (Mojzisch and Schulz-Hardt, 2010). The probability of discovering the relevant knowledge (knowledge of one group member not known to the other group members) to take a high-quality (if optimality criteria exist, also an optimal) decision increases with an increased frequency of information exchange between group members (Wittenbaum et al., 2004). One possible reason for increased knowledge exchange between group members is *group diversity* (in terms of dimensions such as demographic and educational background). The higher the degree of diversity, the higher the probability of higher quality decision outcomes (measured, e.g., in terms of the degree of susceptibility to the framing effect (Yaniv, 2011)). Schulz-Hardt et al. (Schulz-Hardt et al., 2006) report the role of *dissent* in group decision making scenarios: the higher

the dissent in initial phases of a group decision process, the higher the probability that the group manages to share the decision-relevant information. An initial study on selection criteria for preference aggregation in group decision making is reported in Felfernig et al. (Felfernig et al., 2017a) – a major outcome is an observed shift from consensus-based strategies such as *average voting* to borderline strategies such as *least misery* in the case of high-involvement items such as apartments and financial services.

The major focus of our work is to analyze the impact of *recommendation diversity* on the frequency of information exchange between group members. We integrated different recommendation strategies with a varying degree of recommendation diversity into our group decision support environment (Stettinger et al., 2015b) and analyzed the impact of recommendation diversity on knowledge interchange between users. The underlying idea is that too similar recommendations provide only a limited coverage of the whole item space (see, e.g., (McGinty and Smyth, 2003)) and increased diversity helps to introduce new alternatives and to trigger discussions / information exchange with regard to these alternatives.

In contrast to the mainstream in recommender systems research (Jannach et al., 2010; Ricci et al., 2011), we do not focus on improving the prediction quality of recommendation approaches. Our aim is to investigate possibilities to exploit recommendation technologies to foster intended behavior which can also be interpreted as a kind of persuasive technology. In group decision scenarios, it is often more important to increase the performance of the group and foster group members' information exchange, than predicting decisions that will be taken by the group. Based on this idea, we analyze the impact of recommendation diversity on the degree of knowledge exchange in a group. This chapter (extended version of (Felfernig et al., 2015)) analyzes three different basic group recommendation heuristics (aggregation functions) (*min*, *avg*, and *max group distance*) with regard to their impact on the communication behavior (knowledge exchange) within a group.

The major contributions of this chapter are the following. We show that recommendation diversity can help to increase the degree of information exchange in group decision making. Furthermore, a higher degree of information exchange also correlates with a higher preparedness to adapt initially articulated preferences. Finally, we discuss related open research issues.

The remainder of this chapter is organized as follows. In Section 4.3 we introduce the different preference aggregation mechanisms used in our group recommendation approach that help to achieve different degrees of recommendation diversity. Thereafter, in Section 4.4 we introduce the hypotheses and present the results of our empirical study. In Section 4.5 we report open issues for future work. With Section 4.6 we conclude the chapter.

4.3. Preference Aggregation Mechanisms

Different preference aggregation mechanisms were used in our study (see Section 4.4) that was conducted on the basis of our group decision support environment CHOICLA (Stettinger et al., 2015b). This system includes different group preference aggregation mechanisms from social choice theory (Masthoff, 2011) – GD_{min}, GD_{max} and GD_{avg} (see below) have been included for the purpose of the work presented in this chapter. The mentioned aggregation mechanisms differ from each other especially with regard to the calculated diversity (see Formula 4.1). In this context, *diversity(d)* is interpreted in terms of the deviation

of recommendations d (recommended evaluation of specific alternatives, i.e., exam modes) from the evaluations provided by individual group members (eval(u,s) where u is a user and s represents a specific alternative/item, e.g., an exam mode).

$$diversity(d) = \frac{\sum_{u \in Users} |eval(u,s) - d|}{\#Users} \tag{4.1}$$

The following group aggregation mechanisms were used within the scope of our study. First, the *minimum group distance* ($GD_{\min}$) determines a rating d (rating scale [1..5]) that reflects the minimum distance to the individual preferences of the group members (see Formula 4.2). Consequently, Formula 4.2 implements a low-diversity recommendation approach that tries to take into account the initial preferences of group members.

$$GD_{\min}(s) = arg \min_{d \in \{1..5\}} \left(\sum_{u \in Users} |eval(u,s) - d| \right) \tag{4.2}$$

Maximum group distance ($GD_{\max}$) returns a rating d that represents the maximum distance to the preferences of individual group members (see Formula 4.3). Consequently, Formula 4.3 implements a high-diversity recommendation approach that often neglects the preferences of individual group members.

$$GD_{\max}(s) = arg \max_{d \in \{1..5\}} \left(\sum_{u \in Users} |eval(u,s) - d| \right) \tag{4.3}$$

Finally, *average group distance* (GD_{avg}) represents a value between maximum and minimum group distance (see Formula 4.4) and thus can be considered as a compromise between minimum and maximum group distance.

$$GD_{avg}(s) = \frac{GD_{\min}(s) + GD_{\max}(s)}{2} \tag{4.4}$$

These aggregation functions were used as a basis for the user study discussed in the following section.

4.4. Empirical Analysis

Our user study on the impact of different aggregation functions on the preparedness of group members to exchange information has been conducted on the basis of the CHOICLA decision support environment (Felfernig et al., 2015; Stettinger et al., 2015b). A screenshot of the Android version of CHOICLA is depicted in Figure 4.1. N=256 computer science students (12% female, 88% male) participated in the study – all students were enrolled in a software engineering course (object-oriented analysis and design) and assigned to a group that had to implement a software within the scope of the course. Within the scope of our user study, each of these groups also had to choose a *preferred exam mode* for object-oriented analysis and design. An example of such an exam mode is: *1 theoretical question on State Charts (SC), 1 theoretical question on Sequence Diagrams (SD), and two practical exercises on Object-Relational Mapping (ORM).*

All study participants were aware about the fact that there is no guarantee that the preferred exam mode will be taken into account in upcoming exams. The task of each group was to select a specific exam mode on the basis of the CHOICLA decision support environment. Figure 4.1 depicts example screenshots of the

Android version of CHOICLA. The study participants had the chance to choose between n=15 different exam modes which differ (1) in terms of the share of *practical exercises* (PE) and *theoretical questions* (TQ) and (2) in terms of the share of specific topics. For example, PE(2xSC, 2xORM) denotes an exam mode that includes only practical exercises (i.e., no theoretical questions) related to the topics of *state charts* (SC) and *object relational mapping* (ORM).

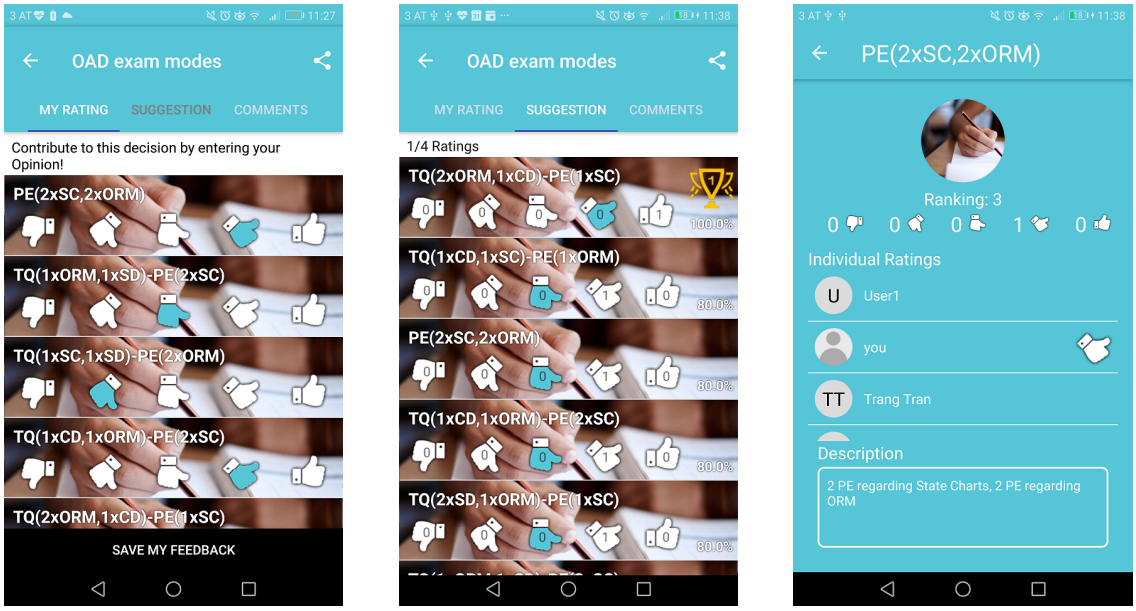

Figure 4.1.: CHOICLA group decision support environment (Stettinger et al., 2015b) (Android version). Recommendations (suggestions) are determined on the basis of the different aggregation functions introduced in Section 4.3.

Within the scope of the study, each group member had to define his/her own preferences with regard to the available exam modes (see Figure 4.1). Before a group member did not define his/her initial preferences, there was no possibility to see the preferences of the other group members (the underlying idea is to avoid anchoring biases that result from a too early preference disclosure (Stettinger et al., 2015a)). On the basis of a short introductory statement before starting the decision process, study participants were encouraged to take a look at the group recommendations (tab *suggestion*) which was done by 91.41% of the participants at least once. An overview of the assignment of individual groups to specific CHOICLA versions that differ in terms of the used aggregation mechanism is depicted in Table 4.2.

aggregation function	#groups	#participants
GD_{min}	17	92
GD_{avg}	12	69
GD_{max}	16	95
total	45	256

Table 4.2.: Assignment of preference aggregation mechanisms to groups.

The hypotheses analyzed within the scope of the empirical study were the following. H1: preference aggregation mechanisms with a higher resulting recommendation diversity increase the degree of knowledge exchange within a group. High-diversity recommendations can act as an anchor (Adomavicius et al., 2011) and can also induce the feeling of dissent and a corresponding need to resolve the dissent. Increased knowledge exchange between group members can increase the probability of identifying the knowledge relevant for taking an optimal decision (Mojzisch and Schulz-Hardt, 2010; Wittenbaum et al., 2004). Examples of the different types of knowledge exchanged within the scope of a group decision processes are shown in Table 4.3. This table summarizes the total amount of messages exchanged between group members that can be assigned to one of the categories of *content-related*, *preference-related*, and *recommendation-related*. In the following we characterize these *categories* on the basis of related examples.

Content-related. A student only took a look at exercises related to a specific topic, e.g., *Object-relational Mapping (ORM)* and asks for further information regarding alternative topics. Another group member points out that there are only a few slides with very simple and understandable examples on the topic of state charts which are also very useful in industrial contexts.

Preference-related. A group member mentions that he/she prefers to include exercises related to the *Unified Process (UP)* compared to *State Charts (SC)*.

Recommendation-related. A participant does not like the group recommendation and he/she wants to discuss assignment topics that are more acceptable for the group as a whole.

Information units exchanged between group members were analyzed manually with regard to the three mentioned categories. In the context of recommendation-related information exchange, we evaluated the valence for recommendation-related comments, i.e., how positive/negative a recommendation was perceived.

H2: a higher degree of knowledge exchange provides more flexibility to change initial preferences afterwards. If more decision-relevant knowledge is exchanged between the members of a group, the amount of global decision-relevant knowledge is increased. This improves the individual capabilities of taking into account additional decision alternatives. Increased knowledge exchange between group members plays a key role to overcome a *discussion bias* (group discussions tend to be dominated by information group members already knew before the discussion (Stasser and Titus, 1985)).

Hypothesis H1 can be confirmed, i.e., the degree of exchanged decision-relevant knowledge depends on the chosen aggregation function. The higher the diversity, the higher the number of exchanged decision-relevant knowledge (see Table 4.3). The number of the given comments for maximum group distance is highest (total number of comments for GD_{max} = 278, GD_{avg} = 92, GD_{min} = 49). Furthermore, also the overall time invested in taking a decision increases with the diversity of recommendations (see Table 4.4).

We can also confirm hypothesis H2. The flexibility of the group members to change their initial preference increases with the higher amount of knowledge exchange. Table 4.5 confirms hypothesis H2 which provides an overview of the changes of initial ratings depending on the supported aggregation mechanisms.

function	content		preference		recommendation		
	#comments	avg.	#comments	avg.	#comments	avg.	valence
GD_{min}	22	1.29	0	0	27	1.59	+4.2
GD_{avg}	31	2.58	26	2.16	35	2.92	+0.9
GD_{max}	79	4.93	91	5.69	108	6.75	-4.4

Table 4.3.: Content-, preference-, recommendation-related comments (#comments, avg. #comments per group, and valence [-5 .. +5] (for recommendation-related comments)).

function	duration (h)		proc. time (min)	
	avg.	std.dev.	avg.	std.dev.
GD_{min}	71.06	13.05	210.71	20.19
GD_{avg}	85.64	26.58	234.56	17.67
GD_{max}	101.18	19.48	278.46	16.74

Table 4.4.: Duration (endtime-starttime) and processing time (total time of system interaction) invested per group for decision task completion (i.e., rating of alternatives).

The average degree of opinion adaptation of groups is highest with GD_{max}.

function	degree of rating adaptation
GD_{min}	0.67
GD_{avg}	1.32
GD_{max}	2.46

Table 4.5.: Changes of initial ratings depending on included aggregation mechanism (difference between original rating and final rating).

Summarizing, the higher the diversity of preference aggregation, the higher the amount of knowledge exchange between group members. Thus, diverse group recommendations can help to increase the probability of identifying optimal solutions due to a higher probability of exchanging knowledge relevant for the optimal decision (Schulz-Hardt et al., 2006; Stasser and Titus, 1985). This can be considered as an important aspect to be taken into account by online decision support environments.

4.5. Future Work

A major focus will be the analysis of further aggregation mechanisms relevant in social choice scenarios (Masthoff, 2011). Of major relevance in this context is to answer the question on the optimal degree of recommendation diversity that helps to optimize the parameters *degree of information exchange* and *perceived recommendation quality*. Tables 4.6 and 4.7 show that the satisfaction with group recommendations decreases with a higher degree of recommendation diversity.

function	GD_{min}	GD_{avg}	GD_{max}
diversity	0.84	1.38	2.23

Table 4.6.: Diversity of group recommendations.

function	very satisfied	satisfied	average	unsatisfied	very unsatisfied
GD_{min}	67	12	9	2	2
GD_{avg}	17	14	12	14	12
GD_{max}	2	1	15	25	52

Table 4.7.: Satisfaction with group recommendations.

4.6. Conclusions

This chapter presents the results of an empirical study that focused on possibilities of increasing the amount of knowledge exchange in group decision scenarios. Thus, in contrast to the mainstream of recommender systems research, we focused on the application of recommendation technologies to improve decision processes per-se. The results of our empirical study show that recommendation diversity has an impact on the frequency of information exchange between group members – the higher the diversity, the more information is exchanged between group members. Furthermore, recommendations with a higher diversity can lead to an increased preparedness of changing initially defined preferences, i.e., these recommendations can be regarded as a mechanism to counteract discussion biases. We regard this work as a contribution to establish recommender systems as a core mechanism to improve the quality of group decision processes.

Chapter 5

Polarization Effects in Group Decisions

Parts of the contents of this chapter have been published in (Atas et al., 2018a). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, data analysis, and evaluation.

5.1. Abstract

Group Recommender Systems aim to support the identification of items that best fit individual preferences of group members. However, decision making behavior of group members can be affected by decision biases which can deteriorate group decision quality. In this chapter, we analyze the existence of *Group Polarization Effects* in two different domains and present a way to counteract these effects. *Group Polarization* is the tendency of a group to make decisions that are more extreme than the average of individual group members' preferences. We analyze *Group Polarization* in the context of *risk analysis* and *cost estimation*. In *risk related* group decisions, we figured out that if individual group members tend to make cautious decisions, then the group decision will be more cautious. However, in decisions related to *cost estimation*, the group estimations are lower than the average of group members' estimations (i.e., cautious shift). Furthermore, our results show that individual group members with diverse preferences are not influenced by *Group Polarization Effects*. The diversity in preferences of individual group members helps to counteract *Group Polarization Effects*.

5.2. Introduction

Recommender systems are decision support systems that support users to identify a set of items fitting their wishes and needs (Jannach et al., 2010; Ricci et al., 2011). Most of the existing recommender systems support single users in identifying useful objects and services such as books, personal computers, songs, and smart phones (Resnick and Varian, 1997). However, there are many scenarios where a group of users receives recommendations and consumes items (Masthoff, 2011; Felfernig et al., 2018d). For instance, *visiting a restaurant* for a dinner with colleagues (Felfernig et al., 2017a), *watching a movie* with friends (Masthoff, 2011), or *planning the next holiday destination* with family (Jameson, 2004).

Similar to single user decisions, group decisions are affected by decision biases. There exist different types of decision biases which can deteriorate group decision quality (Mandl et al., 2011; Felfernig,

2014). *Decoy Effects* (Tversky and Simonson, 1993) influence the decision behavior between two items after a third irrelevant item (i.e., decoy item) is included into the item set. *Serial Position Effects* (Murphy et al., 2006) (also called *Primacy-Recency Effects*) can influence the group decision behavior based on the illustration of recommended items. Experiments show that when recommended items are shown as a list, there is a tendency to recall (i.e., remember) the first and last items (Stettinger et al., 2015b). *Anchoring Effects* (Jacowith and Kahneman, 1995; Adomavicius et al., 2011) negatively influence the group decision behavior when the first piece of information (i.e., anchor element) is shown to other group members. An example of such information can be the *rating or comment of a group member* regarding an alternative item. If such information is shown to other group members, this can affect later perceptions and decisions.

However, there exist many other group decision biases which manipulate the group decision and negatively influence the decision quality. In this chapter, we analyze the influence of *Group Polarization Effects* (Stoner, 1961; Myers and Lamm, 1976; Sunstein, 2002) in two different domains and present a way to counteract these.* James Stoner (Stoner, 1961) discovered *Group Polarization Effects* in 1961 by comparing the risk-taking of individuals and groups. He found out that decisions made by groups are more extreme than the initial decisions of individual members. This phenomenon indicates, if individual group members tend to make risky decisions, then the group decision will be even riskier and if they tend to make cautious decisions, then the group decision will be more cautious. This phenomenon occurs, because if a group of users with the same opinion come together, then, they will most probably follow the same opinion. Lamm et al. (Lamm and Sauer, 1974) analyze *Group Polarization Effects* in a bargaining situation based on a dataset collected in a user study. Each study participant had the task to distribute several units of money (i.e., profit units) between him-/herself and another participant. After comparing the average of individual and group profit units, they observed that group decisions tend to be more extreme than the average of initial inclination of individual members. Another related work is presented in (Sia et al., 2002) which analyzes the impact of *Group Polarization Effects* in computer-mediated discussion forums. The results show that computer-mediated group discussions trigger stronger *Group Polarization Effects* than face-to-face group discussions.

To the best of our knowledge, related work regarding proving and counteracting *Group Polarization Effects* in different domains (i.e., risk analysis and cost estimation) does not exist. In this chapter, we analyze *Group Polarization Effects* in *risk-* and *cost related* group decisions. In *risk related* decisions, we figured out that if individual group members tend to make cautious decisions, then the group decision will be more cautious (i.e., there exist *Group Polarization Effects*). However, if individual group members tend to make risky decisions in the *risk analysis* domain, then the group decision is not riskier. In the context of *cost related* decisions, if individual group members tend to estimate lower costs, then the cost estimation of the group is much lower (i.e., there exist *Group Polarization Effects*). But, if individual group members tend to estimate higher costs, then the cost estimation of the group is not higher than the average of individual group members' estimations. Besides analyzing the impact of *Group Polarization Effects* in different domains, we tried to find a way to counteract this decision bias. We figured out that individual group members with diverse preferences are not influenced by *group polarization effects*, which means that the variety of individual group members' preferences helps to counteract *Group Polarization Effects*.

The remainder of this chapter is structured as follows. Section 5.3 introduces the design of an empirical

*The work presented in this chapter has been partially conducted within the scope of the research projects *WeWant* (basic research project funded by the Austrian Research Promotion Agency) and *OpenReq* (Horizon 2020 project funded by the European Union).

user study and describes the structure of the collected dataset. In Section 5.4, we discuss the results of our evaluation based on the data collected in a user study. Finally, Section 5.5 provides a brief summary of our work, emphasizes the outcome of this chapter, and discusses ideas for future work.

5.3. User Study

We conducted a user study with 211 computer science students where participants were involved in individual- and group decisions in two different domains (*risk analysis, cost estimation*). Our user study was conducted in two phases (*individual decisions, group decisions*) at a university in Austria.[†] *First*, the task of each user study participant was to make decisions *individually* regarding two different domains. In the *next step*, decisions regarding two different domains were made within a group. For group decision making, we formed 211 individual participants (∼89% male and ∼11% female) to 39 different groups (one group with group size of 2, one group with group size of 4, 17 groups with group size of 5, and 20 groups with group size of 6). Before a group decision was initiated, each group member exchanged and discussed his/her opinion with other group members between 5 and 10 minutes. After each group member discussed and articulated his/her opinion, a group decision was made. Furthermore, decisions made by individuals and groups were not performed in a short time. As is well-known, if a decision is triggered right after another decision, the second decision could be influenced by the first decision. In order to prevent the influence of the second decision from the first one, the group decisions have been performed 40 days after the individual decisions. Moreover, all group members had the equal importance within the group and there was not any group leader. Otherwise, group decisions can be influenced by *GroupThink* bias (Esser, 1998; Janis, 1972) which means as soon as the group leader articulates his/her opinion to other group members, group leader's opinion will influence other group members' perceptions and decisions.

5.3.1. *Risk Analysis* domain

In the first domain, a scenario regarding *risk analysis* was described. The description of the scenario was as follows:
"Assume, there is a *recommender Library A* which is applied in several companies and universities for a long time. Unfortunately, *Library A* is based on an already outdated technology which is unable to analyze a huge amount of data. Recently, a new *Recommender Library B* was launched which is much more efficient in dealing with large volumes of data. The new recommender library is therefore well prepared for more demanding tasks in the future. However, the integration of *Library B* involves risks in terms of stability and integration effort."

The question posted to participants was following: *"Please estimate the risk that would makes sense to change to library B?"* The task of each user study participant was to select one out of the following options which represents the taken risk in percentage: 0%, 20%, 40%, 60%, 80%, and 100%, whereby 0% indicates *full risk* and 100% indicates *no risk*.

[†]Graz University of Technology (www.tugraz.at).

5.3.2. *Cost Estimation* domain

In the second domain, participants had to estimate the development costs of the well-known "WhatsApp" chat application (in iOS, Android, and Web) in Euro starting from the first idea to the current version on the open market. Additionally, we mentioned that the developer group of the "WhatsApp" application consists of a large number of employees who work in different areas such as management, testing, software development, and design. The question posted to participants was following: *"How would you estimate the total development costs of the WhatsApp application?"* In this scenario, user study participants had not any predefined options like in the first domain. The cost estimation was not limited and it was freely selectable. Using predefined options in this context could trigger decision biases such as *Anchoring Effects*. Finally, we collected the individual- and group estimations decided by user study participants and analyzed *Group Polarization Effects* based on the following defined hypotheses:

- H1: "*In risk related decisions, groups tend to make decisions that are more extreme than the average of individual group members' preferences.*" It means, if the individual group members tend to make risky/cautious decisions, then the group decisions will be riskier/ more cautious. The related works by Myers and Lamm (Myers and Lamm, 1976), Stoner (Stoner, 1961), and Sunstein (Sunstein, 2002) motivate us to analyze this phenomenon.

- H2:"*In cost related decisions, there is no Group Polarization Effects, because we believe that cost estimations get more realistic (i.e., does not move to the extreme values) if each individual group member first discusses the cost estimation in detail and then makes a group decision.*" The papers of Torp and Klakegg (Torp and Klakegg, 2016) and Futrell et al. (Futrell et al., 2001) state also that an estimation gets more realistic after a group discussion.

- H3: "*If individual group members' preferences are diverse, then group decisions are not influenced by Group Polarization Effects in both domains.*" We think that the group decision will not move to extreme values in situations where group members' preferences are diverse, because a risky or a cautious group decision will always ignore preferences of some group members which will decrease the group satisfaction. To the best of our knowledge, publications with regard to counteracting *Group Polarization Effects* do not exist. This motivate us to develop and analyse this hypothesis.

The evaluation of the collected dataset based on the defined hypotheses is presented in Section 5.4.

5.4. Evaluation and Discussion

For the analysis of *Group Polarization Effects*, we compared whether there is a tendency for a group to make decisions that are more extreme than the average of individual group members' preferences. In order to inspect the impact of *Group Polarization Effects*, we analyzed whether group decisions are riskier, if individual group members tend to make risky decisions. Likewise, we analyzed whether group decisions are more cautious, if individual group members tend to make cautious decisions. Moreover, we analyzed the decision behavior of groups where individual group members have diverse preferences. We believe that group members with diverse preferences help to counteract *Group Polarization Effects*, because if group members' preferences are diverse, then the group decision will not move to the extremes in order to

satisfy every group member. For analyzing *Group Polarization Effects* for risky and cautious groups and to analyze groups with diverse preferences, we divided all groups into the upper-, lower-, and mixed-category (see Figure 5.1).

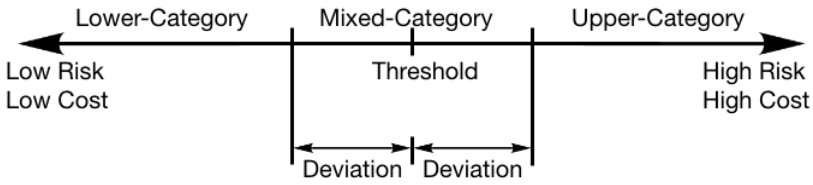

Figure 5.1.: Groups divided into three different categories. Groups in the *Upper-Category* are used to analyze the decision behavior of risky groups, groups assigned to the *Lower-Category* are used to analyze the decision behavior of cautious groups, and groups assigned to the *Mixed-Category* are used to counteract *Group Polarization Effects*.

For analyzing *Group Polarization Effects*, first, we defined a threshold which represents the middle point of preferences and a deviation in order to allocate a space for groups in the mixed-category. If the average of preferences articulated by individual group members is between the $threshold \pm deviation$ and if individual group members' preferences are diverse, then these groups have been assigned to the mixed-category. For instance, if a group consists of four group members and two of them make risky and another two group members make cautious decisions, then the group will most probably make a decision near to the threshold (i.e., middle point) in order to satisfy all group members. Otherwise, groups have been assigned to the upper-category, if the average of individual group members' preferences is higher than $threshold + deviation$ and assigned to a lower-category, if the average of individual group members' preferences is lower than $threshold - deviation$. Groups in the *upper-category* are used to analyze the impact of *Group Polarization Effects* for risky groups and *lower-category* groups are used to analyze the impact of *Group Polarization Effects* for cautious groups. Furthermore, groups assigned to *mixed-category* are analyzed in order to find a way for counteracting this bias.

5.4.1. Group Polarization Effects in *Risk Analysis* Domain

For the first domain (i.e., *risk analysis*), we tried different thresholds (between 40% and 70%) and different deviations (between 1% and 10%) in order to show the polarization effect as clearly as possible. We found out, that the clearest polarization effect occurs for a threshold of 60% and a deviation of 7%. Applying the chosen threshold and deviation leads to following group distribution: 16/39 groups were assigned to the upper-, 7/39 to the lower-, and 16/39 groups were assigned to the mixed-category (see Table 5.1).

We figured out a polarization effect for groups in the upper-category (see Table 5.1). For 13 out of 16 groups assigned to the upper-category, the percentage value articulated by the group was higher than the average of percentages articulated by individual group members. Table 5.1 shows that group percentage values of all 16 groups assigned to the upper-category are higher than the average of individual group members' percentages ($\mu_{individual}$ = 73,42% and μ_{group} = 76,25%). Additionally, we can observe that the standard deviation of percentages for groups in the upper-category is very low ($\sigma_{individual}$ = 3,68% and σ_{group} = 4,06%). This means, there is a cautious shift for groups in the *risk-analysis* domain

Category	# Groups	Individual Preferences			Group Preferences		
		μ	σ	Margin of Error	μ	σ	Margin of Error
Upper-Category	16	73,42	3,68	73,42 $\pm$ 1,86	76,25	4,06	76,25 $\pm$ 2,58
Lower-Category	7	44,95	4,22	44,95 $\pm$ 2,18	71,43	4,51	71,43 $\pm$ 3,61
Mixed-Category	16	60,63	13,23	60,63 $\pm$ 6,14	61,25	5,51	61,25 $\pm$ 3,88

Table 5.1.: Analysis of *Group Polarization Effects* in the *risk analysis* domain for a threshold of 60% and a deviation of 7%. The results are represented in percentage by the mean μ, the standard deviation σ, and the margin of error by using a confidence interval of 95 %. There are 16 groups assigned to the *Upper-Category*, 7 assigned to the *Lower-Category*, and 16 assigned to the *Mixed-Category*. *Group Polarization Effects* occur for groups assigned to the upper-category, but there are no polarization effects for groups assigned to the lower-category. Groups assigned to mixed-category help to counteract *Group Polarization Effects*.

and the percentages are close to the defined threshold, because high percentage value indicates in this context lower risk. However, we did not observe *Group Polarization Effects* for groups assigned to the lower-category. The results show that for only one out of 7 groups, the group percentage was lower than average of individual percentages ($\mu_{individual}$ = 44,95% and μ_{group} = 71,43%). Based on these observations, the hypothesis H1 is partly confirmed, because the analysis of groups in the upper-category confirms the hypothesis H1 and the analysis of groups in the lower-category does not confirm it. Moreover, we figured out that groups assigned to mixed-category (i.e., groups with diverse individual preferences) indicate a solution for counteracting *Group Polarization Effects*. The preferences of group members in the mixed-category indicate that the average of group percentages and average of individual group members' percentages are close to the middle point (i.e., to the threshold) and don't move to the extreme values. Furthermore, we can also clearly see that for groups in the mixed-category, the standard deviation of group preferences is much lower ($\sigma = 5,51\%$) than the standard deviation of individual preferences ($\sigma = 13,23\%$). This means, preferences of individuals were not close to the middle point (i.e., individual preferences were diverse) and group preferences were close to the middle point. Due to this fact, we can conclude that the third hypothesis is confirmed.

5.4.2. Group Polarization Effects in *Cost Estimation* Domain

For the second domain, cost estimations articulated by user study participants regarding the *WhatsApp application* were analyzed. The decisions of user study participants were very different from each other (estimations between 5K € and 2B €). In such cases where values are widely distributed, the comparison of the *cost estimation* made by the group with the average of *cost estimations* made by individual group members makes less sense, because as soon as an individual group member defines a very extreme price (very high or very low price), the average of individuals prices' change significantly. To omit extreme prices articulated by study participants, we clustered the prices with *K-Means Algorithm* using *Java Machine Learning Library* (Abeel et al., 2009). For determining the optimal cluster size, we tried different cluster sizes between 3 and 50 and figured out that the most optimal cluster size is 30. After that, we replaced the prices entered by user participants with the cluster centroid values. This procedure enables us to convert all different prices given by user study participants in 30 different values.

After the clustering step, we tried different thresholds (2,8M € - 500M €) and different deviations (100K € - 2,5M €) in order to show the polarization effect as clearly as possible. We figured out that the clearest polarization effect occurs for a threshold of 3M € and a deviation of 2.2M €. Based on the chosen threshold and deviation, 26/39 groups were assigned to the upper-, 7/39 groups to the lower-, and 6/39 groups were assigned to the mixed-category (see Table 5.2). Groups assigned to upper- and lower-categories were used again to analyze *Group Polarization Effects* and groups assigned to mixed-category were used to counteract *Group Polarization Effects*. The analysis with the chosen cluster size, threshold, and deviation shows that there is no polarization effect for groups assigned to the upper-category ($\mu_{individual}$ = 81,44 million € and μ_{group} = 61,11 million €). However, we identified that *Group Polarization Effects* occurs for groups assigned to the lower-category and figured out that the average of group prices is 60 thousand € (0,24 million € - 0,18 million €) lower than the average of individual prices (see Table 5.2). Additionally, we figured out that all the 7 groups assigned to lower-category have lower group prices than the average of individual prices (i.e., cautious shift). Besides, the standard deviation of the individual- and group preferences assigned to the *Lower-Category* are very low ($\sigma_{individual}$ = 70 thousand € and σ_{group} = 60 thousand €). The analysis of groups in the upper-category confirm our hypothesis H2. However, the analysis of groups in the lower-category does not confirm the hypothesis H2. Therefore, the hypothesis H2 is partly confirmed. Next, we analyzed groups assigned to the mixed-category (i.e., group members with diverse preferences) and show that these groups counteract *Group Polarization Effects*. The price distance between the average of group estimations and the average of individual estimations is 1,69 million € ($\mu_{individual}$ = 2,27 million € and μ_{group} = 3,96 million €). At first sight, the distance of 1,69 million € seems to be huge, but compared to the prices given by participants (5K € - 2B €) this difference is very low. Here, we can again observe that the average of cost estimations made by groups and the average of individual cost estimations are close to the middle point (i.e., to the threshold) and do not move to extreme values. The hypothesis H3 is also confirmed for the 2^{nd} domain.

Category	# Groups	Individual Preferences			Group Preferences		
		μ	σ	Margin of Error	μ	σ	Margin of Error
Upper-Category	26	81,44	5,22	81,44 $\pm$ 3,15	61,11	5,20	61,11 $\pm$ 2,77
Lower-Category	7	0,24	0,07	0,24 $\pm$ 0,03	0,18	0,06	0,18 $\pm$ 0,02
Mixed-Category	6	2,27	1,81	2,27 $\pm$ 0,12	3,96	0,03	3,96 $\pm$ 0,02

Table 5.2.: Analysis of *Group Polarization Effects* in the *cost estimation* domain for a threshold of 3M € and a deviation of 2,2M €. The results of each category are represented in M€ (million Euros) by the mean μ, the standard deviation σ, and the margin of error by using a confidence interval of 95 %. There are 26 groups assigned to *Upper-Category*, 7 assigned to *Lower-Category*, and 6 assigned to *Mixed-Category*. *Group Polarization Effects* occur for groups assigned to the lower-category. Groups assigned to mixed-category help counteract *Group Polarization Effects*.

5.4.3. Discussion

The results of our evaluation regarding *Group Polarization Effects* in *risk-analysis* domain is partly confirmed by (Stoner, 1961; Lamm and Sauer, 1974; Sia et al., 2002). The reason of different outcomes could be the different group sizes which range from 2 to 6, the low engagement of user study participants in the decisions, or the strongly gender biased dataset ($\sim$89% male and $\sim$11% female). To the best of our

knowledge, related work regarding *Group Polarization Effects* in *cost estimations* do not exist. Our results regarding the *cost estimation* domain partly confirm our hypothesis H2 which indicates that cost related decisions are not affected by *Group Polarization Effects*, because cost estimations will get more realistic if each individual group member first discusses the cost estimation in detail and then makes a group decision (Torp and Klakegg, 2016; Futrell et al., 2001). Furthermore, we proved that group members with diverse preferences help to counteract this bias. If a group try to make a decision and if most of group members follow the same opinion, the outcome will be most probably an extreme decision. To counteract this bias, arguments against the group members' opinion have to be provided. These arguments can be provided automatically by a system or a new group member can be added to the group who has different opinion than other group members. This strategy will prevent the group from extreme decisions. Otherwise, the opinion of group members will be always supported and this triggers the *Confirmation* bias (Schwind and Buder, 2012) which is the tendency to search and recall information in a way that confirms one's preexisting beliefs.

5.5. Conclusion and Future Work

With the work presented in this chapter, we have analyzed the impact of *Group Polarization Effects* in two different domains based on a dataset collected in a user study. Furthermore, we presented a way to counteract the group polarization bias in order to prevent the deterioration of decision quality. The analysis of *Group Polarization Effects* in *risk related* decisions shows that group decisions are more cautious if preferences of individual group members are cautious. However, we did not identify *Group Polarization Effects* for risky groups in the *risk analysis* domain. For the second domain (i.e., *cost estimation*), group decisions are not influenced by *Group Polarization Effects* if the individual group members' decisions are over-estimated. However, there is a tendency that the group will make a very low *cost estimation*, if the average of individual group members' estimations are low. Additionally, we confirm in both domains that group members with diverse preferences are not influenced by *Group Polarization Effects*. The variety of individual group members' preferences helps to counteract *Group Polarization Effects*.

With regard to *Group Polarization Effects* we will analyze this bias in further domains such as *personnel decisions* and *requirements prioritization in software requirements engineering* in the future. Another planned future work is to analyze *Group Polarization Effects* based on different genders and age groups and to analyze the impact of this bias in high/low-involvement domain decisions (Felfernig et al., 2017a).

Chapter 6

Socially-Aware Recommendation for Over-Constrained Problems

Parts of the contents of this chapter have been published in (Atas et al., 2018c). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, data analysis, and evaluation.

6.1. Abstract

Group recommender systems support the identification of items that best fit the individual preferences of all group members. A group recommendation can be determined on the basis of aggregation functions. However, to some extent it is still unclear which aggregation function is most suitable for predicting an item to a group. In this chapter, we analyze different preference aggregation functions with regard to their prediction quality. We found out that consensus-based aggregation functions (e.g., *Average, Minimal Group Distance, Multiplicative, Ensemble Voting*) which consider all group members' preferences lead to a better prediction quality compared to borderline aggregation functions, such as *Least Misery* and *Most Pleasure* which solely focus on preferences of some individual group members.

6.2. Introduction

Recommender systems are decision support systems that support users in identifying a set of items fitting their wishes and needs (Jannach et al., 2010). These systems help users to identify useful objects/services (often referred to as *items*) such as movies, books, songs, web sites, financial services, travel destinations, and restaurants (Felfernig et al., 2017a). Nowadays, most recommender systems are designed for single users (Herlocker et al., 2004). However, there are many scenarios where items are supposed to be consumed by groups (Masthoff, 2011). Examples thereof are deciding about a *restaurant* to visit for a dinner with colleagues, deciding about a *movie* to watch with the family, or deciding about a *travel destination* to visit with friends next year.

Group (i.e., socially-aware) recommender systems support the identification of items that to some extent fit the individual preferences of group members. Examples of such systems are the following. O'Connor et al. (O'Connor et al., 2001) present a collaborative filtering movie recommender system for groups

named POLYLENS which is based on the *Least Misery* aggregation function. Masthoff (Masthoff, 2004) introduces concepts to recommend television items to user groups (e.g., choosing a list of television programs to watch together with the family). Moreover, Masthoff found out that group members use some of the aggregation functions (e.g., *Average, Average-Without-Misery*, and *Least Misery*) in television domain and take the fairness aspect into account. Jameson (Jameson, 2004) introduces the TRAVEL DECISION FORUM which is used to elicit and aggregate group members' preferences using *Average* aggregation function for selecting tourist destinations. Ardissono et al. (Ardissono et al., 2003) introduce INTRIGUE which aggregates group members' preferences using *Average* aggregation function and recommends sightseeing destinations to groups. Furthermore, McCarthy et. al (McCarthy et al., 2006) propose a critiquing-based travel recommender system named COLLABORATIVE TRAVEL ADVISORY SYSTEM (CATS) which allows to jointly plan skiing vacations. CATS *averages* the preferences of all group members and recommends a skiing vacation to the group. Ninaus et al. (Ninaus et al., 2014a) introduce a requirement engineering tool named INTELLIREQ, which is tailored to groups of stakeholders designing release plans. INTELLIREQ is based on the *Majority* aggregation function. Stettinger et al. (Stettinger et al., 2015b) introduce a domain-independent decision support environment for groups named CHOICLA which is based on multi-attribute utility theory (Winterfeldt and Edwards, 1986).

In this chapter, we present a constraint-based socially-aware recommender system which recommends digital SLR (Single-Lens Reflex) cameras to groups by applying different aggregation functions.* In this context, we focused on comparing aggregation functions with regard to their capability to predict relevant items in situations where no solutions could be found for a given set of preferences. The result of our study indicates that borderline aggregation functions such as *Least Misery* and *Most Pleasure* have a low prediction quality. They only focus on the lowest and highest values from all individual preferences, i.e., borderline aggregation functions do not consider all existing preferences. On the other hand, consensus-based aggregation functions such as *Average*, *Ensemble voting*, *Multiplicative*, and *Minimal Group Distance* consider the preferences of all group members and are more suitable for predicting a product for groups (see Section 6.5).

The remainder of this chapter is structured as follows. In Section 6.3, we present a working example from the digital camera domain and explain properties and variables of the used dataset. Section 6.4 introduces the clustering approach used to synthesize groups. In Section 6.5, we introduce different group aggregation functions and apply these on the synthesized groups. Subsequently, Section 6.6 presents the results of our evaluation. We discuss issues for future work and conclude the chapter in Section 6.7.

6.3. Working Example

For demonstration purposes, we introduce a group recommendation scenario from the domain of digital cameras. The work presented in this chapter has been conducted in *two phases*: *First*, we conducted a user study where each participant had to declare his/her preferences with regard to a digital camera. *Second*, we synthesized the collected data in order to form groups which were then used to evaluate different preference aggregation functions with regard to their prediction quality.† Each participant of the user study had to declare preferences (i.e., constraints) regarding 10 different camera variables and then select three

*The work presented in this chapter has been partially conducted within the scope of the research projects *WeWant* (basic research project funded by the Austrian Research Promotion Agency) and *OpenReq* (Horizon 2020 project funded by the European Union).

†This approach follows group synthesis approaches as introduced in (Baltrunas et al., 2010)

out of these 10 variables which are most important for him/her. Finally, if the defined preferences were inconsistent with the underlying product catalog, participants had to select an alternative camera from the product catalog. For simplicity, we reduce our product catalog from 20 to 5 entries (see Table 6.1).

ID	eff-res	display	touch	wifi	nfc	gps	video-res	zoom	weight	price
P1	20.9	3.5	yes	yes	no	yes	4K-UHD/3840x2160	3.0	1405	5219
P2	6.1	2.5	yes	yes	no	no	No-Video-Function	3.0	475	659
P3	6.1	2.2	no	no	no	no	No-Video-Function	7.8	700	189
P4	6.2	1.8	yes	yes	yes	no	4K-UHD/3840x2160	5.8	860	2329
P5	6.2	1.8	no	no	no	yes	Full-HD/1920x1080	3.0	560	469

Table 6.1.: An example product catalog. In this context, *eff-res* = effective resolution in mega-pixel, *display* = display size in inch, *touch* = touch screen functionality (yes/no), *wifi* = wireless communication functionality (yes/no), *nfc* = near field communication support (yes/no), *gps* = global positioning system functionality (yes/no), *video-res* = video resolution, *zoom* = zoom factor of the camera, *weight* = weight in grams, and *price* = price in Euro.

All sessions where user requirements were inconsistent (i.e., over-constrained) with the product catalog (i.e., no solution could be found) were stored. In the following, these sessions were used to synthesize user groups that form the basis for our evaluation of different preference aggregation functions. Our approach to group synthesis is introduced in Section 6.4.

6.4. Building Synthetic Homogeneous Groups

Since the dataset from our user study is collected from individual participants, dataset for groups has to be synthesized. Synthesizing a dataset by clustering individual participants (i.e., forming groups) is a common approach in recommender systems (Baltrunas et al., 2010). The metrics in Formulae 6.1 – 6.3 are used to determine preference similarity *sim(p,c)* between a reference participant p who currently declared his/her preferences and a candidate participant c from a set of candidates. In this context, *s(p_i, c_i)* denotes the similarity of the preference i between a reference participant p and a candidate participant c. In addition, *w(i)* indicates the importance (weight) of a preference i for a participant, *val(i)* represents the *value* of preference i, and *minval(p_i)/maxval(p_i)* are minimum/maximum values of a preference i taken from the product catalog.

Formula 6.2, *Near-Is-Better* (NIB) is used if the preferences of the candidate participant have to be as near as possible to the preferences of the reference participant. For instance, the recommended camera should be the one which its price is as near as possible to the price specified by the customer. Formula 6.3, *Equal-Is-Better* (EIB) is applied in situations where the preferences of two participants have to be equal. For a complete overview of related attribute-level similarity functions we refer to (Jannach et al., 2010).

$$sim(p,c) = \frac{\sum_{i \in variables} \mathrm{s}(p_i, c_i) * w(i)}{\sum_{i \in variables} w(i)} \tag{6.1}$$

$$NIB : s(p_i, c_i) = 1 - \frac{|val(p_i) - val(c_i)|}{maxval(p_i) - minval(p_i)} \tag{6.2}$$

$$EIB : s(p_i, c_i) = \begin{cases} 1 & \text{if } p_i = c_i \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

The following part of this section presents the clustering of participants in order to form groups of similar participants (i.e., homogeneous groups). In order to calculate the similarity between two participants (i.e., potential group members), Formula 6.2 (NIB) is applied to the variables *effective resolution, display size, zoom, weight,* and *price*. Furthermore, Formula 6.3 (EIB) is applied to the variables *touch-screen, wifi, nfc, gps,* and *video resolution*. For the purpose of clustering homogeneous group members, the similarity among participants has to be calculated first. Then, the n most similar group members (e.g., cluster size = 4) are selected as group members. This process is repeated for the remaining participants until all groups have been determined. An example of a synthesized group is depicted in Table 6.2.

participant	eff-res	display	touch	wifi	nfc	gps	video-res	zoom	weight	price
1^{st} participant	20.8	2.5	yes	yes	no	yes	4K-UHD/3840x2160	5.0	700	1649
2^{nd} participant	20.9	2.5	yes	yes	no	yes	4K-UHD/3840x2160	5.0	475	2149
3^{rd} participant	20.9	2.7	yes	yes	no	yes	4K-UHD/3840x2160	7.8	560	2749
4^{th} participant	14.2	2.7	yes	yes	no	no	4K-UHD/3840x2160	5.0	475	659

Table 6.2.: A synthesized homogeneous group with cluster size of 4. Participants with similar preferences are considered as group members.

participant	1^{st} imp. variable	2^{nd} imp. variable	3^{rd} imp. variable	chosen product
1^{st} participant	eff-res	weight	price	P1
2^{nd} participant	eff-res	price	video-res	P1
3^{rd} participant	eff-res	display	price	P3
4^{th} participant	eff-res	weight	price	P2

Table 6.3.: The chosen products and three most important camera variables selected by the participants from Table 6.2.

In order to provide a better understanding for the application of similarity metrics, we will show how the similarity between the 1*st* group member and the 2*nd* group member can be calculated (see Table 6.2). For simplicity, we assume equal weights of all the camera variables for both group members ($w(i) = 1$). Formula 6.4 shows the similarity calculation between the 1^{st} group member and the 2^{nd} group member on *effective resolution*. In order to apply Formula 6.2 (NIB) on the *effective resolution* variable, *maximum* and *minimum* effective resolution values are required. The *maxval* and *minval* values are taken from the maximum and minimum effective resolution values of the product catalog (see Table 6.1).

$$
\begin{aligned}
&s(1^{st}participant_{eff-res}, 2^{nd}participant_{eff-res}) \\
&= 1 - \frac{|val(1^{st}participant_{eff-res}) - val(2^{nd}participant_{eff-res})|}{maxval(eff-res) - minval(eff-res)} \\
&= 1 - \frac{|20.8 - 20.9|}{20.9 - 6.1} = 0.9932
\end{aligned}
\tag{6.4}
$$

$$
\begin{aligned}
&sim(1^{st}participant, 2^{nd}participant) \\
&= \frac{\sum_{i \in variables} \mathrm{s}(1^{st}participant_i, 2^{nd}participant_i) * w(i)}{\sum_{i \in variables} w(i)} \\
&= \frac{0,9932 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0,7581 + 0,9006}{1 * 10} \\
&= 0.9652
\end{aligned}
\tag{6.5}
$$

By applying Formula 6.2 and Formula 6.3 to all the remaining variables, the similarity between two group members regarding their preferences can be calculated as presented in Formula 6.5.

The task is now to predict a product for the whole group using aggregation functions (see Section 6.5). The similarities between the group member's preferences and products from the product catalog are presented in Table 6.4.

6.5. Applying Group Aggregation Functions and Recommending Products to Groups

For aggregating individual group member's preferences to a group preference, group aggregation functions have to be used. In collaborative filtering scenarios, group aggregation functions are often applied to the ratings predicted for group members first and then a recommendation for the group can be proposed. However, in our evaluation, we first calculated similarities between group member preferences and products from the product catalog and then applied different aggregation functions on all of these similarities. This intermediate step is needed for determining the attribute similarity. Our approach can be demonstrated based on the group members' preferences in Table 6.2. The group has to decide about the *gps* feature of the camera and group members articulated the following preferences: *eval(*1^{st} *participant, gps) = yes; eval(*2^{nd} *participant, gps) = yes; eval(*3^{rd} *participant, gps) = yes; eval(*4^{th} *participant, gps) = no*. In such a setting, one has first calculate similarities between group member's preferences and products from the product catalog and then apply an aggregation function to derive a group recommendation.

In order to predict products to groups, the following aggregation functions have been applied (see Formulae 6.6 - 6.11). We differentiate between consensus-based (*Average, Minimal Group Distance, Multiplicative,* and *Ensemble voting*) and borderline (*Least Misery and Most Pleasure*) aggregation functions. Formula 6.6 (AVG) returns the *average* value of all individual values for an item *i* as a recommendation, whereby *eval(p,i)* denotes the *evaluation* of item *i* by the participant *p*. Formula 6.7 (*Least Misery*) returns the item with the highest of all lowest individual values and Formula 6.8 (*Most Pleasure*) returns the highest value of all individual values for an item *i* as a recommendation. *Minimum Group Distance* (MGD)

in Formula 6.9 returns a value d which has the minimum distance to all individual values. *Multiplicative* (MUL) in Formula 6.10 returns the product of all individual values for an item i as a recommendation. *Ensemble Voting* (ENS) in Formula 6.11 returns the majority item from the items predicted by individual aggregation functions which are defined in Formula 6.6 - 6.10.

$$AVG(i) = \frac{\Sigma_{p \in Participants} eval(p,i)}{|Participants|} \tag{6.6}$$

$$LMIS(i) = min(\bigcup_{p \in Participants} eval(p,i)) \tag{6.7}$$

$$MPLS(i) = max(\bigcup_{p \in Participants} eval(p,i)) \tag{6.8}$$

$$MGD(i) = arg \min_{d \in \{1..5\}} (\Sigma_{p \in Participants} |eval(p,i) - d|) \tag{6.9}$$

$$MUL(i) = \prod_{p \in Participants} eval(p,i) \tag{6.10}$$

$$ENS(i) = maxarg_{(d \in \{1..5\})} (\#(\bigcup_{p \in Participants} eval(p,i) = d)) \tag{6.11}$$

participant	P1	P2	P3	P4	P5	
1^{st} participant	0.7046	0.5307	0.4179	0.5431	0.4566	
2^{nd} participant	0.7603	0.4809	0.3018	0.5580	0.3816	
3^{rd} participant	0.7311	0.4828	0.3718	0.4708	0.3458	
4^{th} participant	0.4681	0.7673	0.5638	0.6397	0.5339	
aggregation function						predicted product
AVG	**0.6660**	0.5654	0.4138	0.5529	0.4294	**P1**
LMIS	0.4681	**0.4809**	0.3018	0.4708	0.3458	**P2**
MPLS	0.7603	**0.7673**	0.5638	0.6397	0.5339	**P2**
MGD	**0.7046**	0.5307	0.4179	0.5431	0.4566	**P1**
MUL	**0.1833**	0.0945	0.0264	0.0913	0.0322	**P1**
ENS						**P1**

Table 6.4.: Similarities between group member's preferences (see Table 6.2 and Table 6.3) and products from the product catalog (see Table 6.1). Weights ($w(i)$) of Table 6.3 have been taken into account. The weight sequence $\{4,3,2,1\}$ is used here, whereby the first value refers to the 1^{st}, the next value to the 2^{nd}, and the third value to the 3^{rd} most important variable, and the last one refers to the remaining variables. Different aggregation functions are applied on these similarity values, then the product with the highest similarity is recommended to the group.

Table 6.4 depicts the recommended items chosen by aggregation functions based on the similarity values between group member preferences and products from the product catalog. For instance, *AVG* determines *Product 1* (P1) for the whole group. After predicting products for a group, the precision (i.e., prediction quality) of each aggregation function can be calculated (see Table 6.5). The precision of an aggregation function is *the ratio between the number of correctly predicted products and the total number of predictions*

(chosen products).

participant	chosen product	AVG (P1)	LMIS (P2)	MPLS (P2)	MGD (P1)	MUL (P1)	ENS (P1)
1^{st} participant	P1	1	0	0	1	1	1
2^{nd} participant	P1	1	0	0	1	1	1
3^{rd} participant	P3	0	0	0	0	0	0
4^{th} participant	P2	0	1	1	0	0	0
Precision		**50%**	25%	25%	**50%**	**50%**	**50%**

Table 6.5.: Calculated prediction quality (precision) of each aggregation function. The value "1" refers to a correct prediction (i.e., the product chosen by a group member and predicted product by the aggregation function are identical) and "0" refers to an incorrect prediction. Consensus-based aggregation functions *AVG, MGD, MUL,* and *ENS* predict P1 as the most suitable product for the group with a precision of 50% ($\frac{2}{4}$). Borderline aggregation functions *LMIS* and *MPLS* predict P2 as the most suitable product for the group and achieve a precision of 25% ($\frac{1}{4}$).

In the following section we present the evaluation of results based on the data collected from our user study.

6.6. Evaluation

The evaluation of the over-constrained camera dataset collected from our user study will be presented here. We conducted a user study with 263 computer science students (∼85% male and ∼15% female) from two universities in Austria.[‡] Each user study participants articulated 14 different preferences (10 out of 14 preferences refer to different camera variable values, three refer to the three most important camera variables, and a reference value refers to the selected product from the product list). All the 20 products from our product catalog were digital SLR cameras which were manually collected from the NIKON company's website.[§] The acquired data about the importance of the camera variables was essential for our analysis. For instance, a participant who defines the *price* as his/her most important variable is most probably focused on the price and wants get the best digital camera for a specific price. In contrast to a different participant who defined, for example, the *effective resolution* and some other technical properties as the most important ones. Such fine-grained information about the preferences of group members helps to improve the prediction quality of group recommendation.

For determining the aggregation function with the highest precision, we varied different parameters such as *group size* and *importance of the camera variables (weights)*. We analyzed the precision of each aggregation function with 210 different weight sequences and 5 different group sizes (ranging from 2 to 6) and generated 1050 (210*5) combinations thereof. Groups could be formed with different group sizes, for example, for group size of 4, there are 65 groups (263 participants / group size of 4 = 65.75 ⇒ 65 groups). After groups were formed, similarities between group member's preferences and products from product catalog (n=20) were calculated. Then, aggregation functions were applied on these similarities in order to predict a product for a group. We analyzed our dataset with all the combinations and found out that group

[‡]Graz University of Technology (www.tugraz.at) and Alpen-Adria Universität Klagenfurt (www.aau.at).

[§]All the products from the product catalog were manually collected from www.nikonusa.com and www.nikon.de.

size of 4 leads to the highest precision. Consequently, the 4-member groups achieve the highest similarity results for our camera dataset and for group sizes higher and lower than 4, the group gets diverse. In order to confirm this statement, we calculated the average precision of aggregation functions for each group size which is depicted in Figure 6.1. Here we can clearly see, that the precision of aggregation functions increases with group sizes ranging from 2 to 4 and decreases starting with group size 5. We can conclude that there exists a correlation between the degree of group homogeneity and prediction quality of the group recommendation algorithm.

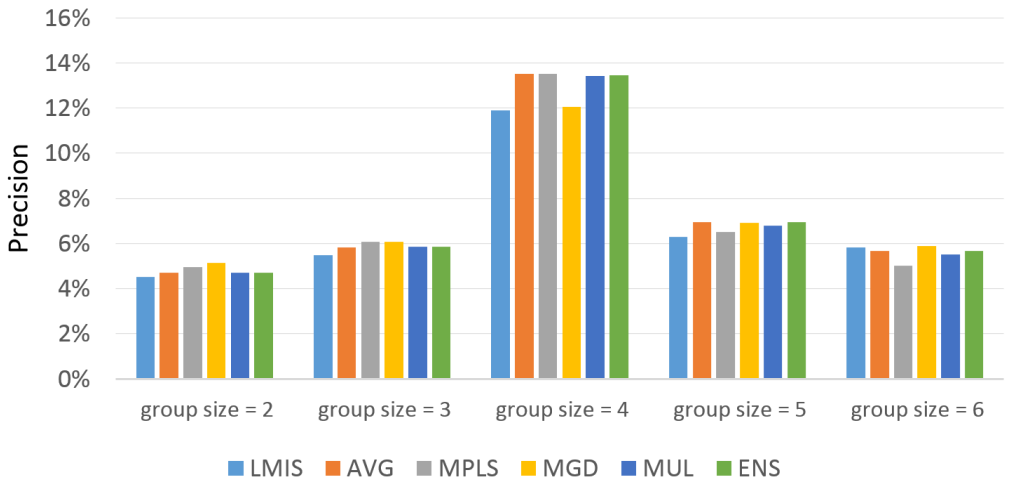

Figure 6.1.: Average precision of aggregation functions for each group size. The highest precision of each aggregation function is achieved for group size of 4.

After the optimal group size was found, we focused on finding the optimal weight sequence (weight sequence which achieves the highest precision). As already mentioned before, 210 different weight sequences were used. Each sequence consisted of 4 different values: the first value refers to the 1^{st}, the next value to the 2^{nd}, and the third value to the 3^{rd} most important variable, and the last one refers to the remaining variables. In our user study, weight sequences were generated by using the following rules:
(1) All the values in the weight sequence should be ranked in the descending order whereby the first value represents the 1^{st} most important variable and the last value represents the less important camera variables.
(2) All values in the weight sequence lie in the range of 1 to 10. Formula 6.12 shows the number of all the used weight sequences (=210).

$$NumberOfWeights = \binom{n}{k} = \binom{10}{4} = \frac{10!}{4! * (10-4)!} = 210 \tag{6.12}$$

In order to determine the weight sequence with the highest precision, we first calculated the average precision of all aggregation functions for each weight sequence and then selected the weight sequence with the highest precision. We figured out that the weight sequence {10,8,4,1} achieves the highest precision (see Figure 6.2). Furthermore, we analyzed the top five weight sequences which achieve the highest precisions in order to find the pattern behind the optimal weight sequences (i.e., weight sequences with the highest precision). One main finding was that the first 2 values which represent the weights of the two most important camera variables are close to each other and the two last values which represent the weight of the 3^{rd} most important variable and the weights of the remaining camera variables are close to each other. In addition to that, regarding the optimal weight sequences, we figured out that the weight

of the most important variable is always as high as possible (i.e., 10) and the weights of the remaining of camera variables are as low as possible (i.e., 1). Therefrom we learn that the domain values of the weight sequences are not equally distributed, rather extreme values are used for the most and least important variables in the weight sequence. Figure 6.2 illustrates the precision of aggregation functions for the weight sequence with the highest precision (i.e., {10,8,4,1}) - the highest precision is achieved again for group size of 4. As aforementioned, depending on the group size, different groups are generated. For instance, for the group size of 3, there are 87 different groups (=263/3). Thereby, the precision of each aggregation function is calculated by taking the average of the precisions generated for all the 87 groups (see Figure 6.2).

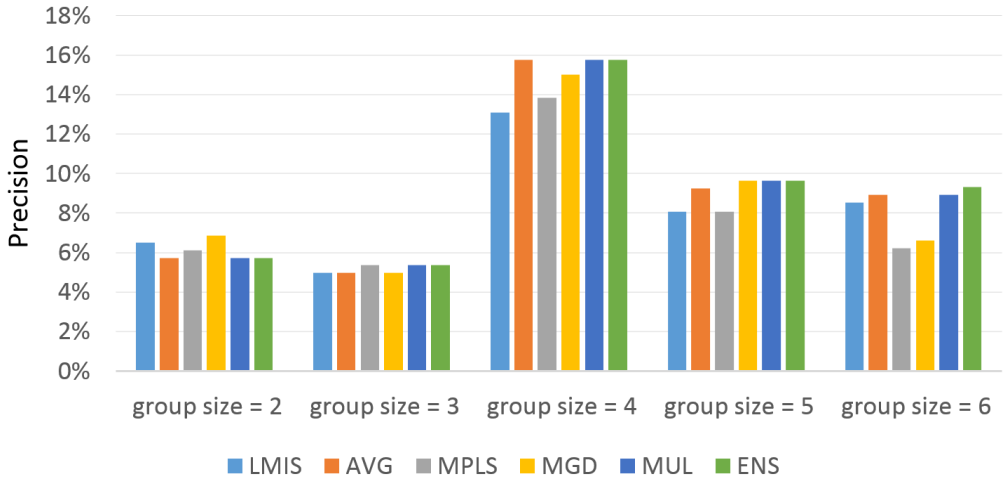

Figure 6.2.: Precision of aggregation functions for the weight {10,8,4,1}. Group size of four leads again to the highest precision for each used aggregation function and aggregation functions *AVG, MUL,* and *ENS* achieve the highest precision.

Once the optimal weight and optimal group size were obtained, we moved our focus towards finding the optimal aggregation function which represents the main task of this chapter. In order to determine the most optimal aggregation function, we took the highest of the average precisions calculated for each aggregation function. The precision of each aggregation function were calculated by taking the average of precisions generated for 1050 combinations (1050 combination is tailored by combining six different group sizes and 210 weight sequences). The final result includes the following precisions of the aggregation functions: *AVG* = 7.34%, *ENS* = 7.32%, *MUL* = 7.26%, *MGD* = 7.22%, *MPLS* = 7.21%, and *LMIS* = 6.80%.[¶] These results are also clearly displayed in Figure 6.1, Figure 6.2, and Table 6.5. They show that *LMIS* and *MPLS* always lead to a low precision since they focus only on the lowest and highest values of all individual preferences and do not consider other group member preferences. Furthermore, *AVG*, *ENS*, *MUL*, and *MGD* which take all of group members' preferences into account lead to better precisions.

[¶]The precision of each aggregation function is not high, because these are the average precisions calculated for all different combinations of weight sequences and group sizes (we varied 210 different weight sequences and six different group sizes).

6.7. Conclusion and Future Work

In this chapter, we analyzed the applicability of preference aggregation functions for groups in situations where the preferences of group members become inconsistent (i.e., over-constrained). Our socially-aware constraint-based recommender system shows that borderline aggregation functions like *LMIS* and *MPLS* lead to a low precision because these functions take the minimum/maximum values from all the individual group member preferences and don't consider the preferences of other group members. Besides, consensus-based aggregation functions (such as *AVG*, *ENS*, *MUL*, and *MGD*) which consider all group members' preferences are more suitable to predict a product for the groups. Furthermore, we also demonstrated that modifications of the group size and the usage of different weight sequences have a potential to achieve better precision. Moreover, we tested the dataset with 210 different weight sequences and different group sizes (2 - 5). We found out that the domain values of the weight sequences are not equally distributed, rather extreme values are used for the most and least important variables in the weight sequence. In addition, the 4-member groups achieve the highest similarity results for our camera dataset and for group sizes higher and lower than 4, the group gets diverse. The chosen group aggregation functions are representatives of consensus-based and borderline functions. The analysis of further related aggregation functions is within the scope of our future work. Socially-aware decisions are usually made in groups consisting of similar group members (e.g. taking vacation with friends or watching a movie with the family). Thus, we clustered similar participants in order to form homogeneous groups. Finally, we plan for the future to build diverse/randomized groups instead of homogeneous groups and try to find the most suitable aggregation function for such groups.

Chapter 7

Socially-Aware Diagnosis for Constraint-Based Recommendation

Parts of the contents of this chapter have been published in (Atas et al., 2019a). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, data analysis, and evaluation.

7.1. Abstract

Constraint-based group recommender systems support the identification of items that best match the individual preferences of all group members. In cases where the requirements of the group members are inconsistent with the underlying constraint set, the group needs to be supported such that an appropriate solution can be found. In this chapter, we present a guided approach that determines socially-aware diagnoses based on different aggregation functions. We analyzed the prediction quality of different aggregation functions by using data collected in a user study. The results indicate that those diagnoses guided by the *Least Misery* aggregation function achieve a higher prediction quality compared to the *Average Voting, Most Pleasure*, and *Majority Voting*. Moreover, another major outcome of our work reveals that diagnoses based on aggregation functions outperform basic approaches such as *Breadth First Search* and *Direct Diagnosis*.

7.2. Introduction

Recommender systems are decision support systems that support users in identifying a set of useful items matching their wishes and needs (Jannach et al., 2010; Felfernig et al., 2017a). Although most existing recommender systems are designed for single user recommendation scenarios (Herlocker et al., 2004), there exist many scenarios where items are supposed to be consumed by groups (Felfernig et al., 2018d; Masthoff, 2011). Examples thereof are deciding on a *restaurant* to have dinner with colleagues, deciding on a *movie* to watch together with family members, deciding on a *travel destination* to visit with friends in the next year, or deciding on a new *digital camera* to use together with your partner.

Compared to collaborative (Goldberg et al., 1992; Konstan et al., 1997) and content-based filtering (Pazzani and Billsus, 1997; Mooney and Roy, 2000), constraint-based group recommender systems (Burke,

2000; Jannach et al., 2010) recommend products and services based on a predefined constraint set. Constraint-based recommender systems are especially applied to complex items and they are also predestined to be used in the context of group decision making, since these systems allow the inclusion of fine-grained constraints (e.g., knowledge-base constraints and requirements of different group members). In order to identify a solution that satisfies all group members, constraint-based group recommenders must take into account the knowledge-base of a domain (e.g., the car domain) as well as the preferences of all group members. These constraint-based systems can be applied to recommend cars (Win and Srisura, 2019), holiday destinations (Torrens et al., 2003), and furniture items to groups (Peischl et al., 2009; Leitner et al., 2012; Reiterer et al., 2015; Reiterer, 2015). The increasing complexity of such items leads to several challenges such as the identification of a suitable recommendation from a huge set of possible solutions/suggestions and the resolution of inconsistencies between the user requirements and the knowledge-base constraints (Reiterer et al., 2015). However, the application of these systems to recommend such complex items in group scenarios introduces further challenges. For instance, in constraint-based group recommender systems (Felfernig et al., 2018d), the inconsistencies between the group member preferences have to be resolved by taking into account the preferences of all group members. When interacting with constraint-based group recommender systems, users usually face situations where the recommender is not able to identify a solution (i.e., a recommendation) for a given set of requirements. In such cases, users have to be supported in finding a way out of the "*no solution could be found*" dilemma (Reiterer et al., 2015). However, if the recommended solution considers only the preferences of some of the group members, the group satisfaction will decrease and thus it can negatively influence the mood of the group. For instance, if a couple commonly decides on the purchase of a new *digital camera*, the final decision must satisfy both persons. Otherwise, this could lead to a situation where at least one of the persons is not satisfied with the new *digital camera*. Therefore, both, the fairness aspect among group members as well as the satisfaction of all group members regarding a commonly consumed item represent the key aspects of a optimal group decision.

To the best of our knowledge, an in-depth analysis with respect to socially-aware diagnosis (i.e., a diagnosis suitable for a whole group of users) for constraint-based recommender systems does not exist. Given this motivation, we developed an approach that determines socially-aware diagnoses guided by aggregation functions in situations where the preferences of group members are inconsistent with the underlying constraint set.* In such situations, a minimal set of group member preferences has to be adapted or deleted. However, the identification of a suitable diagnosis for groups is a cumbersome process. Most approaches related to the determination of diagnoses are based on *Breadth-First Search* and focus on the identification of a diagnosis of *minimal cardinality* (i.e., a diagnosis with a minimal set of constraints) (Felfernig et al., 2004). However, such minimal cardinality diagnoses do not always ensure the most appropriate diagnosis for a group. For instance, if a diagnosis with minimal cardinality only includes preferences of one group member, only this group member has to adapt or delete his/her preferences and the preferences of other group members can remain unchanged. Therefore, such strategies are in most cases not fair and this can decrease the overall group satisfaction. Furthermore, the calculation of diagnoses for an inconsistent constraint set as in *Breadth-First Search* is often infeasible due to an unacceptable runtime performance (Felfernig et al., 2013b). To counteract this problem, a socially-aware diagnosis technique which helps to identify diagnoses that best match the preferences of all group mem-

*The work presented in this chapter has been conducted within the scope of the research projects WeWant (basic research project funded by the Austrian Research Promotion Agency - 850702) and OpenReq (Horizon 2020 project funded by the European Union - 732463).

bers, is required. In this context, we focused on comparing group preference aggregation functions with regard to their capability to predict relevant diagnoses for groups in situations where no solutions can be found. We analyzed group preference aggregation functions such as *Least Misery, Most Pleasure*, *Average Voting*, and *Majority Voting*. The results of our studies show that diagnoses guided by the *Least Misery* aggregation function achieve a higher prediction quality than diagnoses guided by *Average voting*, *Most Pleasure,* or *Majority Voting* aggregation functions. In addition to that, we also compared the prediction quality of aggregation-function-based diagnoses with basic approaches such as *Breadth First Search* and *Direct Diagnosis*. The results indicate that diagnoses guided by group aggregation functions achieve a higher prediction quality than *Breadth First Search* and *Direct Diagnosis*. Moreover, our comparison also includes an analysis of the runtime performance of these algorithms.

The remainder of this chapter is structured as follows. In Section 7.3, we present a working example from the *digital camera* domain. Section 7.4 introduces the calculation of socially-aware diagnoses guided by group preference aggregation functions based on a digital camera recommendation example. In Section 7.5, we explain how the individual users were clustered to generate a dataset for groups using similarity metrics. Section 7.6 introduces different *group preference aggregation functions* which were used for the determination of socially-aware diagnoses. Finally, in Section 7.7, we discuss the results of our evaluation based on the data collected in a user study. We conclude the chapter with a discussion of issues for future work (see Section 7.8).

7.3. Working Example

For demonstration purposes, we introduce a constraint-based group recommendation scenario to be applied in the context of the *digital camera* domain. In this working example, a group of users provides their individual preferences regarding the attributes of a digital camera such as *video resolution, optical zoom*, etc. Due to the natural variance of opinions between different people, some of the provided preferences of the individual group members can be inconsistent. Our constraint-based group recommender system then tries to identify a suitable recommendation (i.e., digital camera) for the group. A constraint-based recommendation task for groups can be expressed as a constraint satisfaction problem (CSP) (Tsang, 1993) which is often used for the definition of constraint-based group tasks (Felfernig et al., 2016).

Definition 1: Constraint-based Group Recommendation Task. A constraint-based group recommendation task can be defined as a triple (V,D,C) where $V = \{v_1, v_2, ..., v_n\}$ is a finite set of variables, $D = \{dom(v_1), dom(v_2), ..., dom(v_n)\}$ represents a set of domains for each variable, and $C = C_{KB} \cup C_R$ conforms to the set of constraints consisting of *knowledge-base specific constraints* (C_{KB}) and *requirement constraints defined by the group members* (C_R). The requirement constraints defined by the group members $C_R = \bigcup C_{R_i}$ is the union of group member requirements where C_{Ri} represents the requirements (i.e., preferences) of group member i.

An example of a constraint-based *digital camera* recommendation task for a group can be expressed as follows. In this example, the variable *video-res* defines the video resolution of the digital camera, the variable *opt-zoom* represents the optical zoom factor of the digital camera, the variable *touch-screen* describes the touch-screen functionality of the display, the variable *water-proof* indicates whether the digital camera is waterproof or not, and the variable *wireless* corresponds to the wireless communication functionality. The

underlying knowledge-base is represented by C_{KB} and the group member requirements are represented by C_R.

- $V = \{$**video-res**, **opt-zoom**, **touch-screen**, **water-proof**, **wireless**$\}$
- $D = \{$
  $dom($**video-res**$) = \{basic$,no-video, Full HD, 4K UHD$\}$,
  $dom($**opt-zoom**$) = \{0.7, 0.85, 1.0\}$,
  $dom($**touch-screen**$) = \{yes, no\}$,
  $dom($**water-proof**$) = \{yes, no\}$,
  $dom($**wireless**$) = \{yes, no\}$
  $\}$
- $C_{KB} = \{$
  c_1 : **water-proof** $= yes \Rightarrow$ **video-res** =4K UHD,
  c_2 : **touch-screen** $= yes \Rightarrow$ **video-res** $\neq basic$,
  c_3 : **opt-zoom** $= 0.7 \Rightarrow$ **video-res** $= basic$,
  c_4 : **opt-zoom** $= 0.85 \Rightarrow$ **video-res** $\neq$4K UHD,
  c_5 : **video-res** $= basic \Rightarrow$ **opt-zoom** $\neq 1.0$
  $\}$
- C_R = $\{$
  $C_{R1} = \{c_6$: **water-proof** $= yes, c_7$: **opt-zoom** $= 0.85\}$
  $C_{R2} = \{c_8$: **video-res** $= basic, c_9$: **wireless** $= no\}$
  $C_{R3} = \{c_{10}$: **touch-screen** $= yes, c_{11}$: **wireless** $= yes\}$
  $\}$

Definition 2: Constraint-based Group Recommendation.
A constraint-based group recommendation for a constraint-based group recommendation task is defined as an instantiation I = $\{v_1 = ins_1, v_2 = ins_2, ..., v_n = ins_n\}$ where $ins_i \in dom(v_i)$. A constraint-based recommendation is consistent if the instantiations in I are consistent with the constraint set $\bigcup c_i \in C$. Furthermore, a constraint-based group recommendation is *complete* if all variables in V are instantiated and *valid* if the constraint-based recommendation is consistent and complete.

7.4. Calculating Socially-Aware Diagnoses

For a set of user requirements which is inconsistent with the underlying knowledge-base, several diagnoses can be found (a diagnosis can be referred to as a minimal set of requirements that needs to be adapted by group members in order to restore consistency). In general, group members do not want to see and evaluate large sets of diagnosis alternatives. The preferred diagnosis should be identified by a diagnosis algorithm and the constraints included in this diagnosis set should be adapted in order to find a suitable solution (i.e., recommendation) for a group (Felfernig et al., 2013b). In this chapter, we identify group-relevant diagnoses by using different preference aggregation functions. In case of an inconsistency between C_R and C_{KB}, a diagnosis has to be determined. In many cases, a *breadth first search* based diagnosis (Reiter, 1987)

is applied in order to find a set of diagnoses $DIAGS = \{\Delta_1, \Delta_2, ..., \Delta_n\}$ with minimal cardinality such that $\forall \Delta_i \in DIAGS : C_{KB} \cup (C_R - \Delta_i)$ is consistent (see the following definitions).

Definition 3: Conflict Set. A conflict set is a set $CS \subseteq \cup C_{Ri}$ such that $CS \cup C_{KB}$ is inconsistent. A conflict set CS is minimal if and only if there does not exist a conflict set $CS' \subset CS$. In our work, the identification of conflict sets is based on the QUICKXPLAIN algorithm (Junker, 2004) which is based on an efficient *divide and conquer* search strategy.

Definition 4: Group-based Requirements Diagnosis. A group-based requirements diagnosis is a set of constraints $\Delta \subseteq C_{Ri}$ such that $C_{KB} \cup (C_{Ri} - \Delta)$ is consistent. A diagnosis Δ is defined as minimal if and only if there does not exist a diagnosis $\Delta' \subset \Delta$ such that $C_{KB} \cup (C_{Ri} - \Delta')$ is consistent.

On the basis of the example in Section 7.3, we identified the following conflict sets: $CS_1 = \{c_9, c_{11}\}$, $CS_2 = \{c_6, c_7\}$, $CS_3 = \{c_8, c_{10}\}$, and $CS_4 = \{c_6, c_8\}$. A conflict can exist between two group members' preferences (e.g., CS_1) or between a certain group member's preferences and knowledge-base constraints (e.g., CS_2). In situations where conflicts exist, a minimal set of constraints that causes conflicts has to be adapted such that at least a solution can be found. For a better understanding, the diagnoses determination of the digital camera example is shown as a HSDAG (Hitting Set Directed Acyclic Graph) (Reiter, 1987) in Figure 7.1. The following diagnoses can be obtained: $\Delta_1 = \{c_9, c_6, c_8\}$, $\Delta_2 = \{c_9, c_6, c_{10}\}$, $\Delta_3 = \{c_9, c_7, c_8\}$, $\Delta_4 = \{c_{11}, c_6, c_8\}$, $\Delta_5 = \{c_{11}, c_6, c_{10}\}$, and $\Delta_6 = \{c_{11}, c_7, c_8\}$.

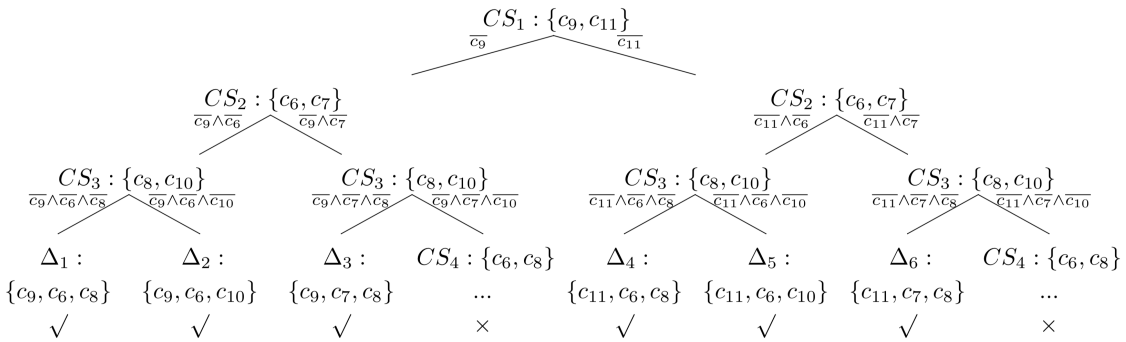

Figure 7.1.: All possible diagnoses of the constraint-based digital camera recommendation example are determined and shown in a *Hitting Set Directed Acyclic Graph*. Thereby, six different diagnoses were found. After one of these diagnoses was applied to the constraint set, a solution for the group could be determined.

A basic approach to determine diagnoses is the *Breadth First Search* (Reiter, 1987) which starts at the tree's root and explores the neighbor nodes first before moving to the next level. The *Breadth First Search* based diagnosis detection for the digital camera recommendation example is depicted in Figure 7.2. It identifies the diagnosis $\Delta_1 = \{c_9, c_6, c_8\}$. The application of Δ_1 leads to at least one solution (e.g., **video-res**=Full HD, **opt-zoom**=0.85, **touch-screen**=yes, **water-proof**=no, **wireless**=yes). As already mentioned before, the calculation and evaluation of all possible diagnoses (see Figure 7.1) or the determination of a diagnosis based on *Breadth First Search* (see Figure 7.2) is often infeasible due to an unacceptable runtime performance (Bangor et al., 2008). Furthermore, a diagnosis determined by the *Breadth First Search*

does not guarantee the identification of the fairest diagnosis for the group. These reasons motivated us to develop a constraint-based recommender system which is able to detect socially-aware diagnoses in an efficient way.

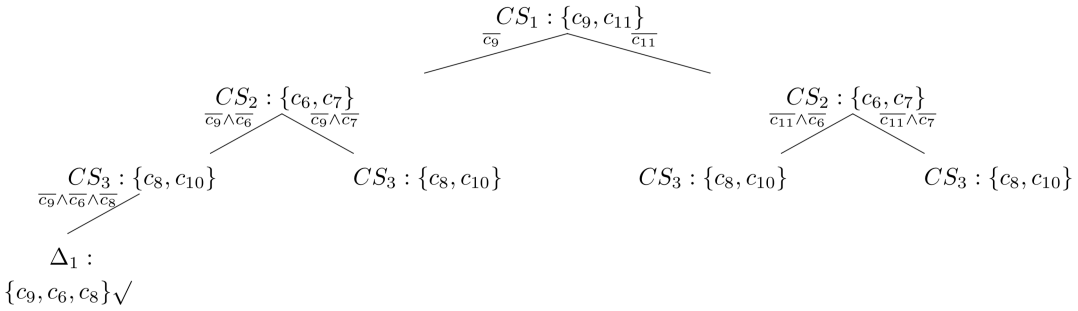

Figure 7.2.: Diagnosis guided by *Breadth First Search* for the constraint-based digital camera recommendation example. The diagnosis $\Delta_1 = \{c_9, c_6, c_8\}$ is determined for the group which leads to at least one solution (e.g., **video-res**="Full HD", **opt-zoom**=0.85, **touch-screen**=yes, **waterproof**=no, **wireless**=yes).

The developed approach determines socially-aware diagnoses which are guided by the following group preference aggregation functions: *Least Misery (LM), Most Pleasure (MP), Average Voting (AVG),* and *Majority Voting (MAJ)*. The determination of a diagnosis guided by group aggregation functions is done in the following ordering: First, a conflict set is chosen (e.g., CS_1) which is calculated by the QUICKXPLAIN algorithm. Then, an aggregation function is used to guide the search (i.e., to resolve the conflict set). A conflict set can be resolved as soon as at least one of the constraints of the conflict set has been deleted or adapted. The HSDAG algorithm represents each of these constraints (included in a conflict set) as a single tree path (see Figure 7.1). Choosing a tree path is equivalent to the deletion of this constraint in a conflict set. The applied group aggregation function (e.g., *Least Misery*) selects an appropriate tree path in HSDAG based on the number of preference adaptations of individual group members. After deciding on a tree path, the QUICKXPLAIN algorithm can be applied again in order to identify another conflict set (e.g., CS_2). The same process is repeated until a diagnosis (i.e., no conflict set) can be identified. The identification of a socially-aware diagnosis for the digital camera recommendation example guided by the *Least Misery* aggregation function is depicted in Figure 7.3. The *Least Misery* aggregation function takes the *fairness* among group members into account and avoids individual misery in the group (Felfernig et al., 2017a). In general, the *Least Misery* aggregation function recommends the item with the highest of all lowest individual ratings (see Section 7.6). However, in this specific context, the number of adaptations instead of the ratings of group members has to be aggregated to a group value. In order to focus on the fairness among group members and to avoid individual misery, *Least Misery* returns the highest adaptation value of all individual values.

For a better understanding of our presented approach, we will explain the identification of socially-aware diagnoses for the digital camera recommendation example guided by the *Least Misery* aggregation function. The first conflict set $CS_1 = \{c_9, c_{11}\}$ identified by the QUICKXPLAIN algorithm can be resolved if we adapt or remove one of the two constraints. The exclusion of the constraint $c_9 : \{$**wireless** = no$\}$ means that the 2^{nd} user would have to delete one of his/her preferences and the exclusion of the constraint

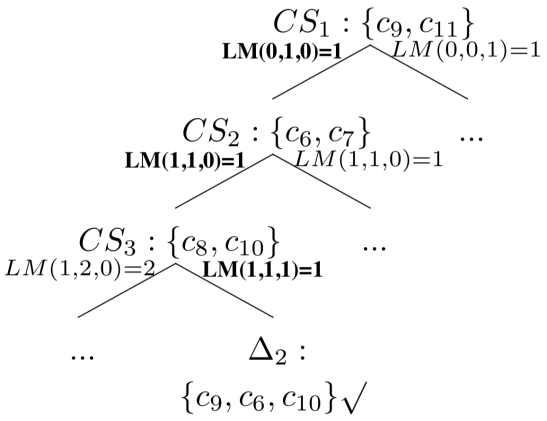

Figure 7.3.: The HSDAG tree of the digital camera recommendation example. The identification of a diagnosis guided by the *Least Misery* aggregation function leads to the diagnosis $\Delta_2 = \{c_9, c_6, c_{10}\}$. The number of adaptations for group members is shown as a triple sequence.

$c_{11} : \{$**wireless** = yes$\}$ means that the 3^{rd} user would have to delete one of his/her preferences. The number of preference adaptations per user is presented as a triple in Figure 7.3, where the first value represents the number of adaptations for the 1^{st} user, the second value refers to the number of adaptations of the 2^{nd} user, and the last value corresponds to the number of adaptations for the last user. This means, the removal of the constraint c_9 leads to the triple *(0,1,0)* and the exclusion of c_{11} to the triple *(0,0,1)*. In the next step, a group preference aggregation function has to be applied in order to guide the search. The application of the *Least Misery* aggregation function for both paths leads to the same aggregated value: *LM(0,1,0)=1, LM(0,0,1)=1*. In this context, the path with the lowest aggregated value (i.e., lowest aggregated number of adaptations) will be chosen, because the lower the aggregated value is, the lower the number of preference adaptations per individual group group member will be. Therefore, choosing the path with a lower aggregated value will satisfy the group more likely than choosing a path with a higher aggregated value. However, if both paths have the same aggregated value, then the left path will be chosen.[†] After the left path (i.e., removing the constraint c_9) has been chosen, the conflict detection algorithm QUICKXPLAIN (Junker, 2004) determines another conflict set which is: $CS_2 = \{c_6, c_7\}$. The exclusion of the constraint c_9 from the previous step and the exclusion of the constraint c_6 (in this step) leads to the triple *(1,1,0)* which finally results in the aggregated group value of *LM(1,1,0)=1*. On the contrary, the removal of the constraints c_7 and c_9 leads to the aggregated group value of *LM(1,1,0)=1*. In such cases, where the aggregated values are equal, the left tree path is chosen again (i.e., c_6 is deleted). After the resolution of both conflict sets CS_1 and CS_2, the conflict detection algorithm QUICKXPLAIN determines another conflict set which is $CS_3 = \{c_8, c_{10}\}$. The deletion of the constraints c_9, c_6, and c_8 (i.e., taking the left path) leads to the aggregated group value of *LM(1,2,0)=2* and the deletion of the constraints c_9, c_6, and c_{10} leads to the aggregated group value of *LM(1,1,1)=1*. Likewise, the removal of the constraint c_{10} results in a lower aggregated value than the removal of the constraint c_8. This ensures that the right path is chosen which appears to be a fair solution for the group. This process is then continuously repeated until a diagnosis for the group can be determined. The use of the *Least Misery* aggregation function leads to the path $\{$*left* $\rightarrow$ *left* $\rightarrow$ *right*$\}$ in HSDAG which is shown in Figure 7.3. This means that the constraints c_9, c_6, and c_{10} have to be adapted or removed from the constraint set in order to identify a suitable digital camera

[†]We used this as a simple tie breaker rule to solve a conflict set.

for the group. The application of the identified diagnosis $\Delta_2 = \{c_9, c_6, c_{10}\}$ to the constraint set will result in a solution for the group (e.g., **video-res**=basic, **opt-zoom**=0.85, **touch-screen**=no, **water-proof**=no, **wireless**=yes).

7.5. Building Synthetic Homogeneous Groups using Similarity Metrics

After the main idea of our approach has been discussed (see Section 7.3 and Section 7.4), some processes to evaluate our approach need to be explained in more detail as well. We tested our approach with a real-world dataset collected in a user study (see Section 7.7). The dataset consists of individual user requirements regarding digital cameras. Due to the fact that all data in our user study has been collected from individual (user study) participants and that our approach is supposed to be applied in group scenarios, our test dataset has to be synthesized to generate a dataset for groups (i.e., groups are synthetically generated). We synthesized a dataset for groups, as our approach identifies diagnoses for groups of users instead of single users. The synthesis of a dataset by clustering individual participants (i.e., forming group of users) is a common approach in the context of recommender systems evaluation (Baltrunas et al., 2010). Usually, most of the group decisions are usually made in groups of users with similar tastes. In order to form groups of such similar users, some similarity metrics were applied (see Formulae 7.1 – 7.3). The term *sim(p,c)* indicates the preference similarity between the participant p who currently declared his/her preferences and a candidate participant c from a set of candidates. The notion *s(p_i, c_i)* denotes the similarity of the variable i between a reference participant p and a candidate participant c. The term *imp(i)* expresses the importance of a variable i for a participant and *val(i)* represents the value of variable i. Moreover, the terms *minval(p_i)* as well as *maxval(p_i)* are minimum and maximum values of a variable i taken from the product catalog. *Equal-Is-Better* (see Formula 7.2) is used in situations where the preferences of two participants must be equal. For instance, Formula 7.2 can be used for the following digital camera variables: *video-res, touch-screen, water-proof*, and *wireless*. Formula 7.3 (*Near-Is-Better*; NIB) is applied if the preferences of the candidate participants have to be as near as possible to the preferences of the reference participant. This equation can be applied for the comparison of numerical (integer and double) variables such as *opt-zoom* of the digital camera. For further information regarding similarity metrics we refer to (Jannach et al., 2010).

$$\mathrm{sim}(p,c) = \frac{\sum_{i \in variables} \mathrm{s}(p_i, c_i) * imp(i)}{\sum_{i \in variables} imp(i)} \tag{7.1}$$

$$\mathrm{EIB} : \mathrm{s}(p_i, c_i) = \begin{cases} 1 & \text{if } p_i = c_i \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

$$\mathrm{NIB} : \mathrm{s}(p_i, c_i) = 1 - \frac{|val(p_i) - val(c_i)|}{maxval(p_i) - minval(p_i)} \tag{7.3}$$

After the collection of the dataset from individual user study participants, similar users were identified and formed to a group by using the aforementioned similarity metrics (see Formula 1-3). For a better understanding of the application of the similarity metrics, we compute the similarity between two example users. We assume that two individual user study participants (i.e., example users) provided the preferences presented in Table 7.1. For the sake of simplicity, we further assume an equal importance of all the digital

camera variables for both users ($imp(i) = 1$). Formula 7.4 shows the similarity calculation between both users with regard to the *opt-zoom* variable by using the *NIB* similarity metric. The *maxval* and *minval* values of the *opt-zoom* variable are taken from the domain definition in the digital camera recommendation example.

User	video-res	opt-zoom	touch-screen	water-proof	wireless
UserA	basic	0.85	yes	yes	no
UserB	Full HD	0.7	yes	no	no

Table 7.1.: Example preferences of two users about a digital camera (UserA and UserB).

$$\begin{aligned} \mathrm{s}(UserA_{opt-zoom}, UserB_{opt-zoom}) &= 1 - \frac{|val(UserA_{opt-zoom}) - val(UserB_{opt-zoom})|}{maxval(opt-zoom) - minval(opt-zoom)} \\ &= 1 - \frac{|0.85 - 0.7|}{1 - 0.7} = 0.5 \end{aligned} \tag{7.4}$$

It is also important to mention that the similarity metrics have to be applied to all product variables (i.e, all variables of the digital camera) in order to calculate the similarity between two users based on their preferences. Formula 7.5 represents the similarity calculation between UserA and UserB.

$$\begin{aligned} \mathrm{sim}(UserA, UserB) &= \frac{\sum_{i \in variables} \mathrm{s}(UserA_i, UserB_i) * imp(i)}{\sum_{i \in variables} imp(i)} \\ &= \frac{0 + 0.5 + 1 + 0 + 1}{1 + 1 + 1 + 1 + 1} = 0.5 \end{aligned} \tag{7.5}$$

The application of Formulae 7.2 and 7.3 on user preferences determines the similarity between two users. After the calculation of the similarity between each user study participant, similar users were clustered in order to form a group. For the generation of groups with similar users, we always picked the first user from the user study participant-list and then calculated the similarity between the first user and all the other users by using similarity metrics. Then, we took the most similar n users and generated a group with a size of $n+1$ (i.e., the first user + n users which were similar to the first user). We repeated the same procedure for the remaining users in order to create several groups with similar users. After the building of groups of similar users, a consistency check based on the group members' preferences was necessary to determine whether there are inconsistencies. In general, the majority of group member preferences contains inconsistencies and these inconsistencies have to be adapted or removed in order to find at least one solution for the group. In our chapter, the inconsistencies between the group member preferences were determined by the conflict detection algorithm QUICKXPLAIN. After the identification of the conflict sets, a diagnosis for the group had to be identified. In our chapter, we identified a diagnosis for the group (i.e., socially-aware diagnosis) by using group preference aggregation functions which are presented in Section 7.6.

7.6. Determining Diagnoses by Applying Group Preference Aggregation Functions

Usually, group preference aggregation functions are applied to aggregate individual group member's preferences (i.e., evaluations regarding an item) (Masthoff, 2011). In this chapter, preference aggregation functions aggregate the number of preference adaptations per individual user to a group value by considering the importance of domain variables from the users' points of view (see Formula 7.6). Formula 7.6 shows that a user's evaluation of a diagnosis fragment *pod (part of diagnosis)* is calculated based on the number of changes and weights (i.e., importance) of variables. In order to guide a diagnosis, the following aggregation functions are used: *Least Misery* (see Formula 7.7) returns the highest value (i.e., number of changes), *Most Pleasure* (see Formula 7.8) returns the lowest value, *Average Voting* (see Formula 7.9) returns the average value, and *Majority Voting* (see Formula 7.10) returns the majority value of all individual evaluations for a diagnosis part *pod* as a recommendation. Finally, Formula 7.11 is applied on a diagnosis part by using one of the group aggregation functions (see Formulae 7.7-7.10) in order to select the tree path which leads to the minimum preference adaptations for the group. To summarize, we can conclude that these aggregation functions are applied to conflict sets in order to guide the diagnosis search.

In group decision scenarios, the preferences of group members often have a different impact on the group decision, due to different importance preferences of the individual group members. For instance, if different family members have to jointly decide on a new house, usually, the preferences of the father and the mother will have a higher impact on the family's decision than the preferences of their children. In such cases, a single preference adaptation for each group member has a different impact on the group decision. If group members with different weights (i.e., importance) exist, then Formulae 7.7-7.10 have to be extended by the importance of users (i.e., **imp(u)*). In our evaluation, we assume that all group members have equal importance and therefore the importance of the individual group members was not considered in the Formulae 7.7-7.10.

$$\text{eval}(u, pod) = \sum_{v \in Variables} (\#\text{changes}(u, pod, v) * \text{imp}(u, v)) \tag{7.6}$$

$$\text{LM}(pod) = max(\bigcup_{u \in Group} \text{eval}(u, pod)) \tag{7.7}$$

$$\text{MP}(pod) = min(\bigcup_{u \in Group} \text{eval}(u, pod)) \tag{7.8}$$

$$\text{AVG}(pod) = \frac{\sum_{u \in Group} \text{eval}(u, pod)}{|Group|} \tag{7.9}$$

$$\text{MAJ}(pod) = majority(\bigcup_{u \in Group} \text{eval}(u, pod)) \tag{7.10}$$

$$\text{selection}(pod) = min(\bigcup_{pod \in pods} \text{eval}(pod)) \tag{7.11}$$

After the determination of a suitable socially-aware diagnosis guided by one of the group aggregation functions, the identified diagnosis is applied to recommend a digital camera to the group.[‡]

[‡]In this context, we recommended only one product (i.e., digital camera) per group.

7.7. Evaluation

In this section, the evaluation of a dataset collected within a user study is presented. The dataset consists of 262 entries and its data have been collected from students at two Austrian universities. The dataset is composed of user requirements regarding digital SLR (Single-Lens Reflex) cameras. Our user study was conducted in three steps. First, each participant of the user study had to declare preferences regarding 10 different variables of a digital camera. After that, each participant selected three out of these 10 variables as his/her most preferred (i.e., most important) variables. In the final step, each participant had to select a camera from the product catalog.[§] Each digital camera in the product catalog was specified by using 10 different variables: *effective resolution* (in mega-pixel), *display size* (in inch), *touch screen functionality* (yes/no), *wireless communication functionality* (yes/no), *near field communication support* (yes/no), *global positioning system functionality* (yes/no), *video resolution*, *zoom factor*, *weight* (in grams), and *price* (in Euro). In our product catalog, we included 20 different SLR cameras from NIKON's company website.[¶] As already mentioned before, we also collected a dataset which includes the importance of the camera variables of each participant which was of high relevance for our analysis. For instance, for some participants the most important variable is the *price*, whereas for some other participants, it might be the *effective resolution* of the camera. These fine-grained details about the individual preferences of the group members allow a reasonable assignment of the participants to suitable groups in the group building process. In addition to that, we also took the importance of the camera variables per participant for the aggregation function based diagnosis determination (see Formula 7.6) into special consideration. This was essential, since the adaptation of a specific camera variable which is necessary to find a solution could otherwise negatively influence the behaviour of the individual participants. For instance, the removal or adaptation of a camera variable which seems to be less important for the participant (e.g., *GPS*) can lead to the following value: $\#changes(userX, pod, gps) * imp(userX, gps) = 1 * 1 = 1$.
However, an adaptation of a very important camera variable from the user's point of view (e.g., *price*) would have a stronger impact: $\#changes(userX, pod, price) * imp(userX, price) = 1 * 4 = 4$.
Although in both cases only a single adaptation (i.e., one change) has been made, the impact will be different (i.e., for userX, the price is 4 times more important than the GPS feature of the camera) because of the different importance of the variables.

In order to take this information into account, we applied different *weight sequences* which represents the importance of the camera variables per participant. A weight sequence consists of 4 different values. An example of a weight sequence is $\{4,3,2,1\}$, whereby the first value refers to the weight of the 1^{st} most important value, the next value to the weight of the 2^{nd} most important value, and the third value to the weight of the 3^{rd} most important camera variable, and the last value represents the weight of the remaining camera variables. As already mentioned before, each participant selected three out of 10 camera variables which are the most important from his or her point of view. Therefore, the *weight sequence* consist of 3 values which represent the importance of the 3 most important variables and one value representing the importance of the remaining 7 variables.

The data collected from our user study was analyzed with $\binom{10}{4} = 210$ different *weight sequences* and 4 different aggregation functions (*LM, MP, AVG,* and *MAJ*). Additionally, we synthesized the dataset in order to group similar participants and we generated groups of 5 different *group sizes* (i.e., the group sizes

[§]The preferences of each user study participant were inconsistent with the underlying product catalog.
[¶]All products were manually collected from `www.nikonusa.com` and `www.nikon.de`

vary from 2 to 6). Consequently, the formation of unequally sized groups leads to a different number of generated groups. For instance, if the *group size* is assumed to be 2, there would be 131 different groups (i.e., $\frac{\#participants}{groupsize} = \frac{262}{2}$). The data collected in the user study was analyzed with 5 different *group sizes* and 210 different *weight sequences* (i.e., $5 * 210 = 1050$ combinations) by using 4 different group aggregation functions. Furthermore, we compared our approach with a *Breadth First Search* based diagnosis detection and a FASTDIAG diagnosis-based algorithm which allows an efficient calculation of a single diagnosis at a time (see Table 7.2).

In order to evaluate and to compare the diagnoses determined by each *group preference aggregation function* as well as by *Breadth First Search* and also by FASTDIAG, the *prediction quality (i.e, precision)* of the diagnosis was determined. As already mentioned before, each user study participant had to choose an alternative camera from the product catalog. In this context, the precision can be considered as the *ratio between the number of correctly predicted products and the total number of all predictions* (see Formula 7.12). The precision of an *aggregation function* is calculated by taking the average of the precision values which were generated for each single group. For instance, for a group size of 2, there exist 131 different groups and the precision represents the average of the precision values of those predictions generated for all 131 groups.

$$precision = \frac{|correctly\ predicted\ diagnoses|}{|predicted\ diagnoses|} \tag{7.12}$$

Group-size	LM	MP	AVG	MAJ	FastDiag	BFS
2	0.126	0.114	0.136	0.114	0.145	0.115
3	0.121	0.113	0.116	0.100	0.113	0.101
4	0.122	0.115	0.115	0.100	0.093	0.095
5	0.124	0.107	0.111	0.112	0.088	0.088
6	0.126	0.097	0.111	0.108	0.086	0.085
average	**0.124**	0.109	0.118	0.107	0.105	0.097

Table 7.2.: Precision of the different *aggregation functions*, the *Breadth First Search* (BFS), and the FASTDIAG algorithm for group sizes between 2 and 6. The last row (*average*) shows the average precision of the different group sizes. The *LM* aggregation function achieves the highest average precision.

The analysis of our dataset shows that the different group sizes do not significantly influence the precision of the applied group preference aggregation functions (see Table 7.2). However, the precision of the FASTDIAG (Felfernig et al., 2013b) and *Breadth First Search* (BFS) algorithms decreases whenever the group size increases. As previously mentioned, the number of group member preferences as well as the number of conflicts increases with an increasing number of group members. Hence, the higher the number of the conflict sets is, the higher the number of possible diagnoses will be. In situations where several diagnoses are available, the most suitable diagnosis has to be determined in an intelligent way. Given a huge number of diagnoses, the probability of finding the most suitable diagnosis for the group using FASTDIAG is very low.[‖] This could be explained by the fact that the FASTDIAG algorithm always tries to identify a diagnosis with minimal cardinality and it does not consider the fairness aspect among group

[‖]In FASTDIAG, the group member preferences have been applied in a randomized ordering. The precision of the FASTDIAG algorithm can be further improved if the group member preferences are ordered based on the importance.

members. Likewise, if the group size is high (i.e., there exist several possible diagnoses), the probability that the most suitable diagnosis is determined by *Breadth First Search* is also very low. Consequently, the precision of the FASTDIAG and *Breadth First Search* algorithms decreases when the group size increases. For the analysis of the dataset, we first started to form groups of size 2 and then progressively increased the *group size* over time. We immediately recognized that different group sizes did not have a significant impact on the precision of the aggregation functions. However, the application of different weight sequences led to different precision results for all group preference aggregation functions. We analyzed the 5 highest (out of 210) *weight sequences* which achieved the highest precision results and noticed that the weight sequences which achieved the highest precision commonly share the same pattern. Also, the values in the weight sequences were not equally distributed. Instead, quite extreme values were used for the most and the least important variables in the weight sequences (e.g., $\{10,6,4,1\}$). This means that the weight of the most important variable should always be as high as possible (i.e., 10) and the weight of the least important variable should be as low as possible (1) in order to reach a high level of prediction quality.

After the analysis of different *group sizes* and *weight sequences*, we could also determine the group preference aggregation function that achieved the highest precision. The precision of each aggregation function was calculated by taking the average of precision values of all 1050 combinations (i.e., group size * number of weight sequences = $5 * 210$). The last row of Table 7.2 represents the average precision of each aggregation function. It indicates that the *Least Misery* aggregation function achieves a higher precision than the *Average voting*, *Most Pleasure*, and *Majority Voting* aggregation functions. Since the analyzed dataset was collected from students, our assumption was that the decision on a *digital SLR camera* for students is always related with high (decision) efforts. On the basis of these facts, we believe that for students, digital SLR cameras fall into the category of *high-involvement* items (i.e., the decision on such items is related with high efforts). The data analysis results mentioned in the paper of Felfernig et al. (Felfernig et al., 2017a) confirm our result and also indicate that group decisions for items in high-involvement domains take the aspect of fairness into account which simply reflects the idea of the *Least Misery* aggregation function. For example, if a family plans to buy a new house (i.e., high-involvement item), the final decision must be fair to everybody and should satisfy all family members. Furthermore, we also analyzed the precision results presented in Table 7.2. These precision results are very low for the following reasons: In our approach, we tried to identify a product which satisfies all group members. However, in our user study nearly every group member (i.e., user study participant) selected a different digital camera when compared to other group members. For instance, if we assume that a group of 4 members selected different digital cameras, the prediction quality could maximally reach up to 25% (100% / 4 group members). This is due to the reason that our approach predicts only one product for the whole group. Furthermore, there are often situations where our approach identifies a digital camera which has not been chosen by any group member (i.e., prediction quality is 0%), because the identified product should satisfy all group members and not only a single participant. Besides, in such cases where the user requirements were inconsistent with the underlying knowledge-base, the user study participants selected a digital camera independent of their preferences. Sometimes the participants selected a product which is not similar to their preferences. This means that the selected product as well as the participants' requirements were not similar. Because of such reasons, the prediction performance results presented in Table 7.2 were very low.

Moreover, in sharp contrast to the *Breadth First Search* algorithm, our approach generates predictions of higher precision quality (see Table 7.2). Furthermore, the *Breadth First Search* based diagnosis detection

did not work efficiently and also consumed a lot of time. For example, the time consumption of *Breadth First Search* for the *group-size*=2 was approximately 4 times higher than the time consumption of our approach and for *group-size*=6, it was even 100 times slower than our approach. The FASTDIAG algorithm detects minimal diagnoses with a logarithmic complexity in terms of the number of consistency checks. Although the algorithm itself works quite fast, the precision of the detected diagnoses is significantly lower than the precision achieved by our approach.

7.8. Conclusion and Future Work

In this chapter, we introduced a novel approach to socially-aware diagnosis by using group aggregation functions. For this purpose, we designed a user study, collected a dataset in a domain of high-involvement items (i.e., digital cameras) and synthesized the dataset from individual participants in order to generate groups of users. Thereafter, a socially-aware diagnosis approach which was guided by group preference aggregation functions, was determined and applied in group member constraints in order to recommend a digital camera to a group. Finally, we measured the prediction quality of each aggregation function and figured out that the *Least Misery* aggregation function achieves a higher precision compared to the *Average Voting*, *Most Pleasure*, and *Majority Voting* aggregation functions. The results show that the aspect of *fairness* plays a major role in selections of high-involvement items based on socially-aware diagnoses. Another important finding was that such group aggregation functions which are based on the determination of a diagnosis achieve a better precision than *Breadth First Search* and FASTDIAG.

Our future plans include the analysis of further group aggregation functions and weight (i.e, importance) sequences. In our work, we clustered similar participants based on the reason that group decisions are usually made in groups consisting of group members with similar tastes. Consequently, the assumption can be made that diverse and randomized groups could be clustered as well as diagnoses for such groups could be determined by using different aggregation functions.

Chapter 8

Towards Similarity-Aware Constraint-Based Recommendation

Parts of the contents of this chapter have been published in (Atas et al., 2019b). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, data analysis, and evaluation.

8.1. Abstract

Constraint-based recommender systems help users to identify useful objects and services based on a given set of constraints. These decision support systems are often applied in complex domains where millions of possible recommendations exist. One major challenge of constraint-based recommenders is the identification of recommendations which are similar to the user's requirements. Especially, in cases where the user requirements are inconsistent with the underlying constraint set, constraint-based recommender systems have to identify and apply the most suitable diagnosis in order to identify a recommendation and to increase the user's satisfaction with the recommendation. Given this motivation, we developed two different approaches which provide similar recommendations to users based on their requirements even when the user's preferences are inconsistent with the underlying constraint set. We tested our approaches with two real-world datasets and evaluated them with respect to the runtime performance and the degree of similarity between the original requirements and the identified recommendation. The results of our evaluation show that both approaches are able to identify recommendations of similar solutions in a highly efficient manner.

8.2. Introduction

Recommender Systems (RS) have become an essential means for guiding users in a personalized way to interesting or useful objects and services (often referred to as *items*) (Jannach et al., 2010; Ricci et al., 2011). These decision support systems help users to identify useful items matching their wishes and needs, such as movies, books, songs, web sites, financial services, travel destinations, and restaurants (Jannach et al., 2010; Gasparic and Janes, 2016; Paraschakis, 2016; Felfernig et al., 2017a). In contrast to traditional recommendation approaches such as collaborative (Konstan et al., 1997) and content-based filtering (Pazzani and Billsus, 1997), constraint-based RS (Burke, 2000; Jannach et al., 2010) recommend products and services based on a given constraint set. These systems are usually applied

in complex domains such as cars, personal computers (PC), and financial services. They allow individual customization of complex industrial products and services in order to satisfy individual customer needs (i.e, user requirements) (Reiterer et al., 2015). When interacting with constraint-based RS, users articulate their requirements (e.g., when interacting with a PC recommender, a user specifies different properties such as *memory type, memory size, processor type, price,* and *brand of PC*). In this context, inconsistent requirements will be automatically or manually adapted (de Kleer et al., 1992; Burke, 2000). Finally, a recommendation will be suggested to the user. A complex product such as a personal computer can have millions of recommendations and the RS has to deal with difficult problems such as system maintainability, consistency maintenance, and efficient response times. One major challenge of constraint-based recommenders is to identify the most suitable recommendation for a user based on his/her articulated requirements. Especially, in cases where user requirements are inconsistent with the underlying constraint set, users have to be supported in finding a way out from the *no recommendation could be found* (Reiterer et al., 2015) dilemma (i.e., identifying a diagnosis). Besides, after identifying and applying a diagnosis, the proposed recommendation should be similar to the user's defined requirements in order to increase the user's satisfaction and be computed with a performance acceptable for interactive settings. The identification of a recommendation which is similar to a user's requirements is a challenging task if millions of recommendations exist. The most naive solution is the comparison of each possible recommendation with a given set of user requirements in order to identify the most similar recommendation for the user which is not possible due to an unacceptable runtime performance.

To the best of our knowledge, such similarity-aware constraint-based RS do not exist. A related work is presented in Eiter et al. (Eiter et al., 2009) where the authors analyse several decision/optimization versions of identifying similar and diverse solutions in the context of *Answer Set Programming* (ASP). The authors introduce offline and online methods to determine the computational complexity of similar/diverse solutions. Hebrard et al. (Hebrard et al., 2005) present a number of practical approaches to identify the distance of similar and diverse solutions in constraint programming and focus on determining the computational complexity of distance functions for similar and diverse solutions. The approach suggested by (Hebrard et al., 2005) calculates the whole set of possible solutions at once and then identifies similar and diverse solutions. In our approaches, we do not calculate the whole set of possible solutions, since this is not feasible due to the high complexity of item domains. Given this motivation, we developed two different approaches that suggest similar recommendations to the users based on their requirements even if the user preferences are inconsistent with the underlying constraint set.* We tested our approaches with two different datasets (PC and bike recommendations) and evaluated them with regard to their runtime performance and the degree of similarity between the original requirements and identified recommendations. The results of our evaluation indicate that our approaches are able to identify similar recommendations with a high similarity degree in a highly efficient manner.

The remainder of this chapter is structured as follows. In Section 8.3, we introduce a working example from the *personal computer* domain. Section 8.4 presents the identification and application of a diagnosis based on users' requirements and the determination of candidate recommendations (i.e., possible solutions). In Section 8.5, we introduce similarity metrics to calculate the similarity between the original requirements and the possible solutions in order to identify the solution with the highest similarity. Section 8.6 introduces

*The work presented in this chapter has been partially conducted within the scope of the research projects *WeWant* (basic research project funded by the Austrian Research Promotion Agency - 850702) and *OpenReq* (Horizon 2020 project funded by the European Union - 732463).

our developed approaches for the identification of similar recommendations. Section 8.7 provides the evaluation results of both approaches with two different real-world datasets. Finally, we conclude the chapter with a discussion of some ideas for future work in Section 8.8.

8.3. Working Example

For demonstration purposes, we introduce a constraint-based recommendation scenario from the domain of personal computers (PC). The example presented in this section introduces the KB (variable definitions and constraints) and user requirements regarding a PC. For simplicity reasons, we used only some of the PC variables as a constraint satisfaction problem (CSP) which is often used for the definition of a constraint-based recommendation task (Tsang, 1993).

Definition 1: Constraint-Based Recommendation Task. A constraint-based recommendation task expressed in CSP representation is defined as a triple (V,D,C) where $V = \{v_1, v_2, ..., v_n\}$ is a set of finite domain variables, $D = \{dom(v_1), dom(v_2), ..., dom(v_n)\}$ refers to the set of variable domains and $C = C_{KB} \cup C_R$ corresponds to the set of constraints representing *product-specific constraints* (C_{KB}) and *requirement constraints defined by a user* (C_R).

A simplified example of a constraint-based recommendation task in the PC domain is the following. In this context, the variable *max-price* represents the maximal price of a PC in Euro, *min-hd-cap* corresponds to the minimum hard-disc capacity in GB, *price* refers to the price of a PC in Euro, *pro-freq* represents the clock-rate of a processor in GHz, *mb-ram-cap* refers to the capacity of the motherboard RAM in GB, and *hd-cap* represents the capacity of the hard-disc in GB. Additionally, there are variables which indicate the importance of PC variables from the user's point of view.[†] For instance, the expression *imp-price = 5* implies that the defined price limit is very important for the user whereas the expression *imp-price = 1* indicates a price limit which is not important from the user's point of view. The product knowledge is represented as $C_{KB} = \{c_1 - c_5\}$ and the user requirements are expressed as $C_R = \{c_6 - c_{13}\}$.

- $V = \{$**max-price**, **min-hd-cap**, **price**, **pro-freq**, **mb-ram-cap**, **hd-cap**, **imp-price**, **imp-hd-cap**, **imp-pro-freq**, **imp-mb-ram-cap**$\}$
- $D = \{dom($**max-price**$) = \{1000, 2000, 3000, 3500\}$,
  $dom($**min-hd-cap**$) = \{512, 1024\}$,
  $dom($**price**$) = [400 ... 3500]$,
  $dom($**pro-freq**$) = \{2, 2.2, 2.6, 3.15\}$,
  $dom($**mb-ram-cap**$) = \{8, 16\}$,
  $dom($**hd-cap**$) = \{64, 128, 256, 512, 1024\}$,
  $dom($**imp-price**$) = dom($**imp-hd-cap**$) =$
  $dom($**imp-mb-ram-cap**$) = dom($**imp-pro-freq**$) = [1 ... 5]\}$
- C_{KB}=$\{$ c_1: (**hd-cap**$\geq$512 && **mb-ram-cap**$\geq$8) $\Rightarrow$ **price**$\geq$2250,
  c_2: (**hd-cap**$\geq$1024 && **mb-ram-cap**$\geq$16) $\Rightarrow$ **price**$\geq$3250,
  c_3: **mb-ram-cap**=16 $\Rightarrow$ **pro-freq**$\geq$3.15,

[†]If this information is not provided, equal importance of all variables is assumed.

c_4: **price**$\leq$**max-price**,
c_5: **min-hd-cap**$\geq$**hd-cap**$\}$

- $C_R = \{c_6$: **max-price** $= 2000, c_7$: **min-hd-cap** $= 1024, c_8$: **pro-freq** $= 2.6$,
  c_9 : **mb-ram-cap** $= 16, c_{10}$: **imp-price** $=3, c_{11}$: **imp-hd-cap** $=2$,
  c_{12} : **imp-mb-ram-cap** $=5, c_{13}$: **imp-pro-freq** $=1\}$

A constraint-based recommendation can be defined based on the given constraint-based recommendation task.

Definition 2: Constraint-based recommendation. A constraint-based recommendation for a recommendation task is defined as an instantiation I = $\{v_1 = ins_1, v_2 = ins_2, ..., v_n = ins_n\}$ where $ins_i \in dom(v_i)$. A constraint-based recommendation is *consistent* if the instantiations in I are consistent with the $\bigcup c_i \in C$. Furthermore, a recommendation for a constraint-based recommendation task is *complete* if all variables in V are instantiated and *valid* if the recommendation is consistent and complete. In this chapter, we ranked our solutions and always recommend the first solution (i.e., recommendation) to the user. For a detailed discussion at ranking approaches for solutions we refer to (Winterfeldt and Edwards, 1986).

8.4. Identification of Personalized Diagnoses

For the recommendation task introduced in Section 8.3, it is not possible to find a solution due to some inconsistencies between the user requirements C_R and the product-specific constraints C_{KB}. For instance, user's constraints regarding the *clock-rate of the processor* and *RAM capacity of the motherboard* (c_8, c_9) contradict each other, because the third constraint in the knowledge base (KB) indicates that the *clock-rate of the processor* must be greater or equal to the value of 3.15 GHz if the *RAM capacity of the motherboard* is 16 GB. Consequently, we have to identify a minimal set of user constraints which has to be adapted or deleted in order to get rid of the *no recommendation could be found* dilemma. In some certain cases where the user requirements C_R are inconsistent with the underlying constraint set C_{KB}, the users have to be supported in identifying constraints which trigger an inconsistency. Such constraints can be determined on the basis of the minimal conflict detection principle (Junker, 2004). On the basis of minimal conflict sets, diagnoses (i.e., hitting sets) can be determined thereof (Reiter, 1987). Such diagnoses are proposals of requirements which should be changed such that the system is able to find a solution.

Definition 3: Conflict Set. A conflict set is a set $CS \subseteq C_R$ such that $CS \cup C_{KB}$ is inconsistent. A conflict set CS is minimal if and only if there does not exist a conflict set $CS' \subset CS$. In the working example defined in Section 8.3, there exist two minimal conflict sets: $CS_1 = \{c_6, c_7\}$ and $CS_2 = \{c_8, c_9\}$. CS_1 and CS_2 are conflict sets since each individual conflict set is in conflict with C_{KB}. A basic approach to determine minimal diagnoses from minimal conflict sets is the so-called *hitting set directed acyclic graph* (HSDAG) (Reiter, 1987).

Definition 4: Diagnosis. A diagnosis defines a set of constraints $\Delta \subseteq C_R$ such that $C_{KB} \cup (C_R - \Delta)$ is consistent. A diagnosis Δ is defined as minimal if and only if there does not exist a diagnosis $\Delta' \subset \Delta$ such that $C_{KB} \cup (C_R - \Delta')$ is consistent. Based on the identified minimal conflict sets, the following diagnoses can be identified by using the HSDAG approach: $\Delta_1 = \{c_6, c_9\}$, $\Delta_2 = \{c_6, c_8\}$, $\Delta_3 = \{c_7, c_8\}$, and $\Delta_4 = \{c_7, c_9\}$. A basic approach for resolving conflicts is to adapt or to delete (see Section 8.6) the constraints contained in a diagnosis set. In order to identify the most suitable diagnosis for the user, the importance of diagnosed

constraints (i.e, diagnosed variables) from the user's point of view has to be taken into account. For instance, selecting the first diagnosis ($\Delta_1 = \{c_6, c_9\} = \{\textit{max-price} = 2000\ €,\ \textit{mb-ram-cap} = 16\ GB\}$) would require the deletion or adaptation of the constraints regarding *max-price of PC* and the *RAM capacity of motherboard* variables, but as indicated in Section 8.3, the *price of PC* is *moderately important* (imp-price=3) and the *RAM capacity of motherboard* is very important (imp-mb-ram-cap=5) for the user. The deletion or adaptation of important user constraints decreases the user satisfaction. In such cases where several diagnoses exist, one should select the diagnosis which contains unimportant user constraints such that users are still satisfied with the proposed constraint adaptation. Moreover, this strategy helps to identify recommendations which are similar to users' requirements, because the similarity calculation (between user requirements and identified recommendations) applied in our approach depends on the importance of the variables (see Section 8.5). This means, the adaptation of important variables will often deteriorate the degree of similarity compared to the adaptation of less important variables. For an automated minimal diagnosis detection, we apply the FASTDIAG (Felfernig et al., 2013b) algorithm, which allows an efficient calculation of one diagnosis at a time. Furthermore, FASTDIAG enables to identify a minimal diagnosis which consists of unimportant user requirements. If the user requirements provided to FASTDIAG are already sorted based on their importance, then the algorithm tries to identify a minimal diagnosis which consists of unimportant constraints (i.e., user requirements) in the first half of the constraint list. In our working example, Δ_3 contains less important constraints compared to Δ_1, Δ_2, and Δ_4. The application of $\Delta_3 = \{c_7, c_8\} = \{\textit{min-hd-cap} = 1024\ GB,\ \textit{pro-freq} = 2.6\ GHz\}$ leads to six different solutions. One of the six possible solutions is the following:
price = 1000 €, **pro-freq** = 3.15 GHz, **hd-cap**=64 GB, **mb-ram-cap**= 16 GB

Our approaches (see Section 8.6), take the first 10 solutions and recommend the solution to the user which is most similar to the user's requirements. The similarity between the user's requirements and the identified solutions can be calculated using similarity metrics presented in Section 8.5.

8.5. Determination of Similarity Degree Using Similarity Metrics

Similarity metrics (McSherry, 2004) are applied for the similarity calculation between the user requirements and a recommendation (see Formulae 8.1- 8.5). The metrics are denoted as *more-is-better* (MIB; e.g., hard-disc capacity of a PC), *less-is-better* (LIB; e.g., price of a PC), *nearer-is-better* (NIB; e.g., clock-rate of the processor should be as near as possible to 2.6 GHz), and *equal-is-better* (EIB; color of a PC) (McSherry, 2003). The term *sim(r,u)* indicates the similarity between a recommendation r from a set of recommendations and requirements of a user u. The notation $s(r_i, u_i)$ represents the similarity between the requirement of user u and the recommendation r with respect to the variable i (attribute-level similarity). In addition, *imp(i)* denotes the importance of a variable i from the user's point of view and *val(i)* represents the *value* of variable i. The terms $minval(r_i)$/$maxval(r_i)$ are minimum/maximum values of a variable i taken from the KB definition.

$$sim(r,u) = \frac{\sum_{i \in variables} s(r_i, u_i) * imp(i)}{\sum_{i \in variables} imp(i)} \tag{8.1}$$

$$MIB : s(r_i, u_i) = \frac{val(u_i) - minval(r_i)}{maxval(r_i) - minval(r_i)} \tag{8.2}$$

$$LIB : s(r_i, u_i) = \frac{maxval(r_i) - val(u_i)}{maxval(r_i) - minval(r_i)} \tag{8.3}$$

$$NIB : s(r_i, u_i) = 1 - \frac{|val(u_i) - val(r_i)|}{maxval(r_i) - minval(r_i)} \tag{8.4}$$

$$EIB : s(r_i, u_i) = \begin{cases} 1 & \text{if } r_i = u_i \\ 0 & \text{otherwise} \end{cases} \tag{8.5}$$

8.6. Approaches for the Identification of Similar Recommendations

We developed two approaches for the identification of recommendations which are similar to the user's requirements. The *soft relaxation-based approach* is based on a soft relaxation of inconsistent user requirements. A soft relaxation in this context makes a strictly specified user requirement less strict. For instance, a soft relaxation for a user constraint *price* = 100 € can be represented as a deviation of 10 €: $\{price \geq 90 € \;\&\&\; price \leq 110 €\}$. This relaxation strategy is used to identify items which are similar to the users' specifications. In contrast to a soft relaxation, a hard relaxation does not use any deviation, it simply deletes the specified value instead. The *hard relaxation-based approach* identifies similar recommendations by deleting the diagnosed user requirements (= hard relaxation) and using search heuristics from CHOCO constraint solver.

Soft Relaxation-Based Approach: The first developed approach for the identification of similar recommendations is based on a soft relaxation of inconsistent user requirements. This approach uses following lines of the Algorithm 1: 1-4, 10, and 12-17. It takes at first user's requirements and checks whether the user's requirements are consistent with the underlying KB by using a constraint solver (see lines 1-2 in Algorithm 1). If they are consistent, the recommender will only consider the first 10 solutions and recommends the solution which has the highest similarity (see line 13). As a KB of complex items such as cars, PCs, and smart homes consisting of hundreds of constraints, the user's requirements are often inconsistent with the underlying KB. In the case of an inconsistency, a suitable diagnosis has to be identified. For identifying a suitable diagnosis, the user's requirements will be sorted by their importance. Thereafter, the FASTDIAG algorithm analyzes the already sorted user constraints and identifies a diagnosis (see line 3). After the identification of a suitable diagnosis, all variables in the diagnosis set will be relaxed (soft relaxation) in order to find solutions similar to the specified user requirements (see line 4). The goal of the relaxation is to avoid empty search results. Such strategies try to identify solutions which are similar to the user's requirements. The presented relaxation strategy is a basic approach for identifying similar numerical values (Dabrowski and Acton, 2011). For the relaxation of non-numerical variables (e.g., *color* of the PC), there also exist some strategies. Wilson and Martinez (Wilson and Martinez, 1997) present improved versions of heterogeneous distance functions for nominal variable values by representing the variables as vectors. This means that, variables are represented by different aspects (e.g., the colors are presented in a RGB color model). Another approach to relax non-numerical variables is the relaxation based on the popularity (i.e., the most popular variable value will be used). This strategy can help to identify recommendations where the user's requirements are inconsistent with the underlying KB, but it is not able to identify recommendations similar to the user's requirements. For instance, if the user specifies that the color of the PC should be *white* ($\{c_1$:*color*=*white*$\}$), a relaxation of *color*=*black* does not make sense, even if the *black* color is popular. We apply another simple approach for non-numerical variables. If

a non-numerical user requirement was inconsistent with the underlying KB, then its neighbor values from the set of variable domains were used for the relaxation. For instance, assuming that a user specifies that the color of the PC should be *yellow* which would be inconsistent and that the domain of the color-variable is defined as follows: dom(**color**)= {black, brown, red, orange, yellow, green, blue, gray, white}. In such cases, we are choosing the neighbors of the user's specified value and relax the non-numerical user requirement as follows: $\{color = yellow \parallel color = orange \parallel color = green\}$. After the relaxation of the inconsistent variables, a recommendation will be suggested to the user (see line 10). Thereafter, the similarity metrics mentioned in Section 8.5 are applied in order to calculate the similarity between the user requirements and the recommended item. Finally, the average of all similarities for all users is taken into account in order to determine the quality of the *soft relaxation-based approach*.

Hard Relaxation-Based Approach: This approach identifies recommendations similar to the user's requirements by deleting the diagnosed user requirements (i.e., hard relaxation) and by using constraint solver heuristics. This approach uses the following lines of the Algorithm 1: 1-3, 6-8, and 12-17. First, the approach takes the user's requirements and checks whether the user requirements C_R are consistent with the underlying KB C_{KB}. In the case of a consistency, the recommender will only consider the first 10 solutions and recommend the solution which has the highest similarity with the original user requirements (see line 1-2 in Algorithm 1). Otherwise, FASTDIAG will only consider all those user requirements which are already ordered with respect to their importance and identify a diagnosis with minimal cardinality. The identified diagnosis will most probably contain less important user requirements which is a strategy to prevent a deterioration of the user's satisfaction. Thereafter, all diagnosed user requirements will be deleted from the user constraint set which guarantees that at least one item can be recommended based on the user's remaining constraints. After that, the variable- and value-ordering heuristics of the CHOCO (Prud'homme et al., 2017) constraint solver are applied in order to identify similar recommendations.[‡] In CHOCO, the user defines constraints and tries to identify solutions which satisfy his/her requirements by using alternating constraint filtering algorithms with a search mechanism. The following CHOCO heuristics are applied in our approach:

- **CHOCO value-ordering heuristics:**
  IntDomainMax, IntDomainMin, IntDomainMedian, IntDomainRandom, IntDomainRandomBound, IntDomainMiddle
- **CHOCO variable-ordering heuristics:**
  FirstFail, Largest, Smallest, Random, AntiFirstFail, MaxRegret

At the beginning, a variable-ordering heuristic has to be selected for the application of the CHOCO heuristics, to determine the ordering of the variables. Then, a value-ordering heuristic can be applied for each variable to determine the ordering of the values. There are 36 different heuristic combinations (6xVariable- and 6xValue-Ordering heuristics). The application of a heuristic combination will not affect the recommendation list, but its application will lead to a different ranking of the list. Our goal is to identify the heuristic combination which leads to a recommendation list where the most similar recommendations are located on the top of the list. However, we did not apply CHOCO value-ordering heuristics for each variable, because for some variables the used value-ordering heuristics should not change. For instance, for the *price of the PC*, only the *less-is-better (IntDomainMin Value ordering heuristic)* metric makes sense

[‡]CHOCO (Prud'homme et al., 2017) is a free open-source constraint solver library for the Java programming language. http://www.choco-solver.org/

(i.e., the cheaper the PC is, the higher the user's satisfaction would be). Therefore, for some variables, the applied value-ordering heuristic will not change and for the remaining variables, we tried all the heuristic combinations. The heuristic combination which leads to the most similar recommendations is later used to test the *hard relaxation-based approach* (see line 8). Finally, the rest of the algorithm will be executed as explained before.

Algorithm 1 Identification of similar requirements based on soft- and hard-relaxation

```
1:  for user u : users do
2:    if checkConsistency(u.reqs) == false then
3:      diagVars = FASTDIAG(orderReqsBasedOnImp(u.reqs))
4:      relaxReqs = reqsRelaxation(diagVars)
5:      if checkConsistency(u.reqs + relaxReqs) == false then
6:        deleteDiagConstraints(diagVars)
7:        applyChocoHeuristics()
8:        rec = getRecommendation(u.reqs - diagVars, heuristic)
9:      else
10:       rec = getRecommendation(u.reqs + relaxReqs)
11:     end if
12:   else
13:     rec = getRecommendation(u.reqs)
14:   end if
15:   similarityPerUser += calculateSim(rec, u.reqs)
16: end for
17: similarity = similarityPerUser / users.size()
```

8.7. Evaluation

The evaluation of similarity-aware constraint-based recommendation based on both approaches is presented in this section. The training and testing of our approaches are based on two different knowledge bases (from *personal computer* and *bike* domains) defined by the *Configuration Benchmarks Library* (CLib).[§]

8.7.1. Personal Computer Dataset

The first dataset represents the KB of a *personal computer* which consists of 45 variables with different domain values and of more than 200 KB constraints. Such knowledge bases from complex domains have usually millions of solutions and the similarity calculation between the user requirements and all possible solutions in order to identify the most similar recommendation is not possible due to the poor runtime performance. For testing our approaches, we artificially generated 500 random user requirements.[¶] Additionally, we also randomly generated the importance of variables from the user's point of view.

[§] https://www.itu.dk/research/cla/externals/clib/, Maintained by CLA group. KB definition in CSP representation: https://github.com/CSPHeuristix/CDBC/

[¶] All user requirements were inconsistent with the underlying KB.

Soft Relaxation-Based Approach: The result of this approach achieves a very high similarity on average, but it does not always guarantee to find recommendations for all users (see Table 8.1). However, in certain cases where the recommender can identify a solution, the similarity degree will be very high since all the variable values of the identified recommendation will be close to the requirements defined by the user. The non-cumulative normal distribution of the similarities is depicted in Figure 8.1. As shown in Table 8.1 and Figure 8.1, the average of similarities based on the *soft relaxation-based approach* is very high and data points are close to the mean (μ = 94,11 % and σ = 2,66).

Hard Relaxation-Based Approach: In this approach, we train the system with 500 automatically generated user requirements in order to identify the most suitable CHOCO heuristic. We figured out that the *Largest* variable-ordering as well as the *IntDomainMedian* value-ordering heuristic combination achieve the highest similarity on average. This means that the CHOCO constraint solver orders variables based on the largest values in its domain and then selects the median value from the variable domain. After the identification of the most suitable heuristic combination, we tested the same approach with another 500 user requirements which were not used in the training phase.

The results show (see Table 8.1 and Figure 8.1) that the mean and the standard deviation of the *soft relaxation-based approach* are significantly better than the mean and the standard deviation of the *hard relaxation-based approach*. However, an average similarity of 84,68 % and a standard deviation of 9,91 % is also an acceptable result. Moreover, this approach is able to identify recommendations for all 500 users, whereby the *soft relaxation-based approach* identifies recommendations only for 82 out of 500 users. The time consumption of both approaches is about 20 seconds which means that a recommendation per user can be calculated in ∼40 ms which is quite acceptable.[‖] The main reason for the huge time consumption in both approaches is the CHOCO constraint solver which creates a new CHOCO model for each user. A new CHOCO model will be generated for each user which takes all the user's requirements and KB constraints into the account.

8.7.2. Bike Dataset

The second dataset represents the KB of a *bike recommendation* which consists of 34 variables with different domain values and more than 350 KB constraints.[**] Examples for constraints can be *frame-type, color,* or *tire-height* of the bike. Furthermore, there are also constraints regarding the customers (i.e., users) such as *gender, height,* and *weight* of a customer.

Soft Relaxation-Based Approach: The evaluation on both datasets shows similar results (see Table 8.1 and Figure 8.1). The results show that the average similarity of the *soft relaxation-based approach* is very high and the data points are very close to the mean. Furthermore, the time consumption of the *soft relaxation-based approach* using the *bike* dataset is much higher than the *personal computer* dataset. The reason for this is the high number of KB constraints in the *bike* dataset (120 ms vs. 40 ms per recommendation).

Hard Relaxation-Based Approach: We can from the results (see Table 8.1 and Figure 8.1) observe that

[‖] Our approaches were implemented in programming language Java and were executed on a computer with following properties: Windows 10 Enterprise; 64-bit operating system; Intel(R) Core(TM) i5-5200 CPU @ 2,20 GHz processor; 8,00 GB RAM.

[**] For training and testing our approaches, we automatically generated again 500 user requirements. All user requirements were inconsistent with the underlying KB.

the *hard relaxation-based approach* is able to work independently from the domain. The application of this approach on both datasets leads to similar results. As already discussed, the time consumption of the *hard relaxation-based approach* using the *bike* dataset takes longer than using the *personal computer* dataset (84 ms vs. 40 ms per user recommendation).

Relaxation Type	μ **in %**	σ **in %**	**Margin of Error in %**	**Confidence Interval in %**	**Number of Recommendations**	**Time in sec.**
Soft (PC)	94,11	2,66	94,11 $\pm$ 0,08	95	82/500	23,2
Hard (PC)	84,68	9,91	84,68 $\pm$ 0,04	95	500/500	19,4
Soft (Bike)	91,24	7,02	91,24 $\pm$ 0,08	95	190/500	60,2
Hard (Bike)	81,31	16,17	81,31 $\pm$ 0,06	95	500/500	42,1

Table 8.1.: Similarity results of the *soft-* and *hard relaxation-based approaches* on both datasets. μ indicates the mean and σ the standard deviation.

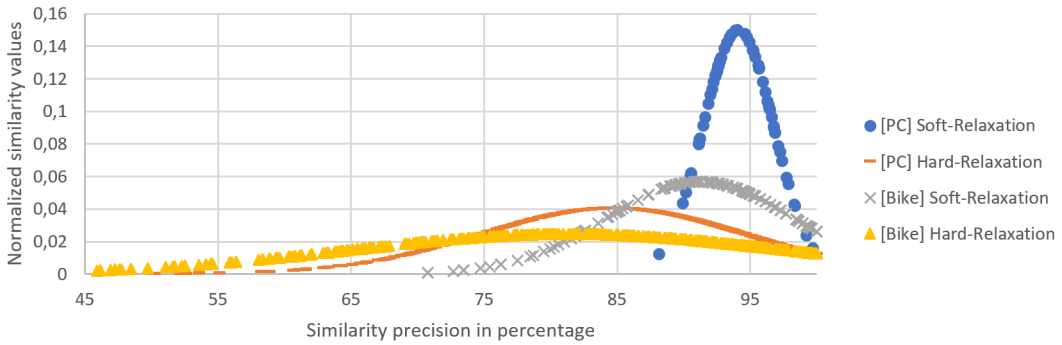

Figure 8.1.: Normal distribution of similarities on *PC* and *bike* datasets using *soft-* and *hard relaxation-based* approach.

Our observation of the characteristics of each approach using datasets from different domains leads to the conclusion that both approaches are able to identify similar recommendations independently from the domain even when the user's requirements are inconsistent with the underlying KB. The *soft relaxation-based approach* is able to identify similar recommendations with a high similarity ($>$ 91 %), but it is not able to identify recommendations for all the users. The *hard relaxation-based approach* can identify similar recommendations for all users, but its average similarity ($>$ 81 %) is lower than the average similarity of the *soft relaxation-based approach*. To properly counteract these undesired issues, we propose a hybrid approach which combines the advantages of both approaches (see Algorithm 1). The hybrid approach simply tries to identify a recommendation using the *soft relaxation-based approach*. Whenever a recommendation can be identified, the similarity will be very high. Otherwise, the *hard relaxation-based approach* will be applied which guarantees that at least one recommendation can be suggested to each user.

8.8. Conclusion and Future Work

This chapter analyzed two different constraint-based RS which can recommend items similar to the user's requirements. Both recommendation approaches are able to identify similar recommendations even when the user's requirements are inconsistent with the underlying KB. We evaluated both RS in terms of the similarity degree and the runtime performance with KB from different domains and figured out that both approaches are able to recommend similar items in an effective and efficient way. Finally, we propose a hybrid recommender system which can identify similar recommendations with a high degree of similarity by combining the advantages of both approaches.

With regard to *similarity-aware constraint-based recommendations*, we want to evaluate our approaches in other domains such as *cars*, *round trips*, and *smart homes* in order to determine their performance. Moreover, we plan to develop our own heuristic based on the idea of the CHOCO heuristics in order to identify recommendations which are similar to the user's requirements. Another idea regarding future work is to develop intelligent relaxation strategies for numerical and non-numerical variables in order to recommend similar items to the users.

Chapter 9

Automated Identification of Type-Specific Dependencies Between Requirements

Parts of the contents of this chapter have been published in (Atas et al., 2018b). The author of this thesis provided major parts of this chapter in terms of writing, literature research, user study design, and evaluation.

9.1. Abstract

Requirements Engineering is one of the most important phases in a software project. The elicitation of requirements and the identification of dependencies between these requirements appears to be a challenging task. In this chapter, we present an approach to automatically identify requirement dependencies of type requires by using supervised classification techniques. Our results indicate that the implemented approach can detect potential requires dependencies between requirements (formulated on a textual level). We evaluated our approach on a test dataset and figured out that it is possible to identify requirement dependencies with a high prediction quality. We trained and tested our system with different classifiers such as *Naive Bayes, Linear SVM, k-Nearest Neighbors,* and *Random Forest*. The results show that *Random Forest* classifiers correctly predict dependencies with an F_1 score of $\sim$82%.

9.2. Introduction

A software project can be regarded as successful if the defined goals are achieved. *Requirements Engineering* plays a crucial role for the success of software projects. Incomplete Requirements Engineering processes mostly lead to project failures (Hofmann and Lehner, 2001; Mobasher and Cleland-Huang, 2011; Felfernig et al., 2017c). Leffingwell (Leffingwell, 1997) reports that 40% of project failures are caused by poorly defined software requirements. Bokhari et al. (Bokhari and Siddiqui, 2010) indicates that the specification of a requirement should be *complete, correct, consistent, unambiguous, verifiable,* and *traceable*. In order to prevent project failures, each core activity in the Requirements Engineering process has to be tested and reviewed in detail by experts. Core activities of a Requirements Engineering process are *the definition and elicitation of requirements, the negotiation of requirements, the identification of dependencies, the identification of stakeholders, quality assurance*, and *release planning*. Since the size and complexity

of software projects increases rapidly, there is a high demand on applying automated and intelligent techniques to support core activities of the Requirements Engineering processes (Castro-Herrera et al., 2009; Mobasher and Cleland-Huang, 2011; Ninaus et al., 2014b).
Dependency Identification is a crucial activity in Requirements Engineering. During this phase, stakeholders have to find and define correct dependencies very carefully and as early as possible (Li et al., 2012). In the case of *incomplete*, *incorrect*, or *inconsistent* dependencies, the project can most probably not be successfully completed. If dependencies between requirements are manually defined by stakeholders, there is a certain risk that the defined set of dependencies is incomplete and may contain conflicts. To phrase it differently, dependencies play a major role in the detection of inconsistencies in a software project. Only if stakeholders are aware of all existing inconsistencies early in a project, effortful re-designs and re-implementations can be avoided. Nowadays, many projects have a large number of requirements and the identification of dependencies can be challenging for humans. Due to this fact, there is an urgent need for automated technologies which can assist the stakeholders in finding dependencies between requirements. This is exactly where our approach comes into play. It empowers stakeholders to counteract the aforementioned issues by supporting them in finding dependencies. Up to now, some related work exists regarding the extraction of pieces of legal texts from documents and the detection of references between them (Tran et al., 2014; Sannier et al., 2017). In particular, in the context of Requirements Engineering several content-based techniques are exploited to identify similar requirements based on a textual-level (Ninaus et al., 2014b). However, all of these approaches lack the ability to predict the direction of a dependency.

Our main contribution in this chapter is to introduce an intelligent system in order to tackle the open issues regarding dependencies between requirements by using supervised learning techniques based on text-mining. We introduce an intelligent approach to automatically identify requirement dependencies of type requires.* Overall, there exist different dependency types such as *requires, excludes,* and *includes*. Our approach focuses on detecting requirement dependencies of type *requires* since it is the dependency type that most frequently occurs in software projects (Ferber et al., 2002). Moreover, due to its nature, this type of dependency is directional. Thus, it has the highest impact on the assignment of requirements to releases as well as on the ordering of the requirements in release planning scenarios. By taking also the direction of the *requires*-dependency into account, dependency detection is able to work even more precisely. Also, the input that is suitable for dependency detection has to be selected more carefully. This also introduces an additional strictness aspect when designing dependency detection mechanisms. Furthermore, considering the direction of the *requires*-dependency when predicting a dependency is a more challenging task than just predicting the existence of a dependency between two requirements regardless of the *dependency*-direction. In other words, the direction of the *requires*-dependency for a given requirement-pair must be exactly predicted by the system in order to be considered as a correctly classified dependency. For example, assuming that R_1 requires R_2 and R_2 does *not* require R_1, a predicted *requires*-dependency for the pair (R_1, R_2) would be considered as a correctly classified example. However, a *requires*-dependency for the pair (R_2, R_1) would be considered as an incorrectly classified example.

In order to be able to predict such dependencies, our system uses a document classification approach. Based on the *title* and the textual *description* of the requirements, dependencies between them are identified using *Natural Language Processing* (NLP) techniques (Ryan, 1993). Additionally, different *preprocessing* techniques are applied in order to prepare a dataset for the classifiers and to filter irrelevant

*The work presented in this chapter has been conducted within the scope of the Horizon 2020 project OPENREQ (732463).

information. We analyzed our approach with different classifiers such as *Naive Bayes, Linear SVM, k-Nearest Neighbors,* and *Random Forest*. The results show that the *Random Forest* classifier can correctly detect new dependencies between requirements with a F_1 score of $\sim$82%.

The remainder of this chapter is organized as follows. In Section 9.3, we present the design of our empirical study. Section 9.4 introduces the applied *Natural Language Processing* techniques. Subsequently, in Section 9.5 we present the results of our evaluation. Section 9.6 describes the threats to internal and external validity. Finally, Section 9.7 provides a brief recapitulation of our work, emphasizes the outcome of this chapter, and discusses ideas for future work.

9.3. User Study

The evaluation of our approach is based on a real-world dataset collected within a user study that focused on the detection of dependencies between requirements for a *sports watch*. The dataset contains text-based requirements which have been defined in cooperation with industry partners (software development companies) engaged in one of our research projects. It is important to point out that the text of the requirements has not been written for the purpose of this study, but for the purpose of having a complete real-world basis for the development of a sports watch which can be used in different kind of application scenarios. This means that the text of the requirements does not contain certain sections which clearly indicate the existence of a dependency to another requirement (e.g., "*requires ...*", "*is dependent on ...*"). Hence, the requirements collection can be considered as not being influenced by undesired bias effects and therefore represents a good basis to be used to evaluate a dependency-detection approach once all dependencies have been found.

The purpose of the user study was to enrich the given collection of requirements with all dependencies existing between these requirements. This was achieved by letting the participants find these dependencies as well as by including additional dependencies found by a selected team of requirements engineering experts with longstanding experience. This way, a very complete final dataset was generated (see Section 9.4.1). However, before the students' dependencies could be combined with the expert's dependencies, the dependencies found by the students had to be further reviewed by a different team of experienced requirement engineers in order to assure a well-defined and correct set of the students' dependencies. The final dataset was used to train and validate our system in order to automatically identify dependencies between requirements. Within the scope of the user study, dependencies between requirements were collected by 182 computer science students at a university in Austria the Graz University of Technology.[†] The user study was conducted in two steps. *First*, we showed 30 different requirements regarding a *sports watch* to each participant. Each requirement consists of an *id*, a *title*, and a textual *description*. The title and the description were both written in German language. A few examples of such requirements (translated to English) are listed in Table 9.1. In order to counteract the undesired occurrence of *Serial Position Effects* (Murphy et al., 2006; Stettinger et al., 2015b), all requirements were shown in random order to each participant.
In the *second* step, the set of randomly ordered requirements was shown to participants. The participants were asked to manually find all correct dependencies of type *requires* between two requirements based on the shown title and description. Thereby, the participants stated 6461 dependencies in total, whereby 657 of those dependencies were unique. For instance, considering all the requirements from Table 9.1, the

[†]Graz University of Technology (`http://www.tugraz.at`).

Id	Title	Description
R1	Speed Measurement	As evaluation after a workout, the average speed must be shown. The following statistics should be displayed: average speed and maximum speed. For measuring the average and maximum speed, time and distance have to be measured, and a storage unit for storing the data is necessary.
R2	Distance Measurement	For statistical purposes, a distance measurement is necessary which needs data from a GPS sensor. This data is needed for the evaluation software and therefore stored in memory.
R3	GPS	To capture position data, a GPS sensor should be used. Through the measured position and time information, the speed and the distance can be measured.
R4	Ideal BMI	Based on the data on height, weight, body fat, age and gender, the watch should be able to calculate the ideal BMI for a user.
R5	Infrared	In order to be able to connect the watch with a computer, WLAN, Bluetooth, and infrared modules must be available.

Table 9.1.: Five example requirements out of 30 requirements which have been shown to participants of the user study.

following dependencies of type *requires* can be found:

- $R_1 \rightarrow R_2$
- $R_1 \rightarrow R_3$
- $R_2 \rightarrow R_3$

For example, the dependency of "$R_1 \rightarrow R2$" indicates that the requirement with the title "Speed Measurement" (R_1) *requires* the requirement with the title "Distance Measurement" (R_2). After collecting all the dependencies, *Natural Language Processing* techniques have been exploited to support the automated detection of dependencies (see Section 9.4).

9.4. Approach to Automated Dependency Detection

For the training and testing of our system, we used several classification approaches (see Section 9.4.3). Based on these approaches, the system automatically learns a model which then can be used to identify new dependencies. Before training our system, *preprocessing of the input data* (i.e., data collected from the user study participants) was necessary in order to clean and prepare the data for the succeeding feature generation phase. First, we extracted the title and the description of each requirement and split them into words (also known as *tokens*). After this so-called *tokenization process*, *stop words* (e.g., "the", "a", "in") and punctuation/special characters (e.g., "-", ".", ";") were removed in order to filter out those tokens containing non-relevant information. This step was crucial as it could negatively influence the overall performance of the used classifier. Next, we determined and assigned corresponding POS-tags (*Part-Of-Speech tagging*) using the *Stanford CoreNLP* library (Manning et al., 2014) to all tokens. Furthermore, we applied *Lemmatisation* by using the *Pattern library*[‡] on the complete list of tokens based on the assigned

[‡]Pattern Library: `https://www.clips.uantwerpen.be/pattern`

POS-tags. Thereby, the lemmatisation technique aims to replace a word with its lemma (Plisson et al., 2004). For instance, words like "comes", "coming", and "came" will be mapped to the same word "come". This significantly reduces the number of words (i.e., tokens) and, hence, also reduces the number of features which were used for training. This way, the trained model is less complex, avoid over-fitting scenarios and hence can behave more flexible.

9.4.1. Dataset

Based on the 30 predefined requirements (see Section 9.3), there are 870 possible dependency combinations (more formally, $\binom{30}{2}.2$), whereby each dependency can only exist between two different requirements. The *binomial coefficient* has to be multiplied with the factor *two* since a dependency between two requirements of type *requires* cannot be considered as bidirectional. This is due to the fact, that there are two different dependency possibilities of type *requires* between each pair of requirements. For instance, the dependency "R_1 *requires* R_2" (more formally, $R_1 \rightarrow R_2$) is not the same as "R_2 *requires* R_1" (more formally, $R_2 \rightarrow R_1$). However, in practice, a dependency between $R_1 \rightarrow R_2$ can exist in parallel to its reverse equivalent dependency $R_2 \rightarrow R_1$. For instance, in order to know when the battery of a sports watch is charged, the *charging function* requires a *charging indicator/display*. Likewise, *charging indicator* without a *charging function* is useless and therefore *charging indicator* requires a *charging function*.
We created a pool of all 870 possible dependency combinations between two different requirements and assign a boolean value to all of these combinations which indicates the existence of a dependency between the two requirements. Whenever a dependency between two different requirements was identified by a user study participant, the boolean value *true* was assigned to this dependency, otherwise the value *false*. We call this boolean value the *existence-label* and used it to train and test our model (see Section 9.4.3). We figured out, that the existence label lacks of a huge class imbalance between the two classes *false* and *true* as the number of requirement-pairs which are not dependent, dominated. In total, we observed that 764 of all 870 requirement-pairs were independent in terms of *requires*-dependency (i.e., the *false* class represents the *majority class*) and the remaining 106 of all 870 pairs were dependent in terms of *requires*-dependency (i.e., the *true* class represents the *minority class*). As already mentioned in Section 9.3, the user study participants identified 657 unique dependencies of type requires. However, to train and test our approach, we only used 106 out of 657 dependencies. Hence, the set of 106 dependencies in combination with the complete set of 30 requirements represents the final dataset that was used to train and test the system. The dependencies of the final dataset were obtained by combining a given set of dependencies found by experts with a set of dependencies found by the participants of the user study (see Section 9.3). Those dependencies (manually) detected by the study participants were further reviewed and included in the final dataset by a team of selected experts with longstanding experience and practical knowledge in the field of requirements engineering. The purpose of combining the dependencies found by the experts and the study participants was to derive a *golden record* which can assure a higher sensitivity and accuracy of our trained model. The *golden record* then further serves as a high-quality *ground truth* to the model which ensures completeness, preciseness, and clearness of the data. Consequently, by providing a profound/stable *ground truth*, the trained model can make more accurate predictions. This is very important since a weak ground truth could make the model unable to predict certain dependencies and hence lead to a poor prediction quality. For example, if we would have included only the dependencies found by the experts and would have ignored the dependencies found by the students in the final dataset, our model would have been trained with dependencies which are incorrect or incomplete. This could make the model quite unstable in terms of prediction quality and lead to falsely classified dependencies. In particular, the model would

be unable to detect some (actual) dependencies. As a (fatal) consequence, such undetected (i.e., unseen) dependencies could further cause a requirements engineering project to fail.

Moreover, in order to train and to build a reliable classification model which can distinguish between all possible classes and hence perform well on unseen data, also the balancing of all classes (in this particular case we only have two classes *true/false*) is an indispensable task. Otherwise, a high overall prediction quality could be easily achieved even if the system did not identify any of the existing dependencies, since the majority class is represented by *false* (i.e., no dependency between the two requirements of a requirement-pair exists). Due to this fact, random undersampling was applied in order to re-balance the pool of all dependencies (Bowyer et al., 2011; Galar et al., 2012). This way, we can make sure that 50% of all selected *requirement*-pairs are dependencies (i.e., *true*) and 50% of all selected pairs are *not* dependencies (i.e., *false*). For the analysis of our approach in training- and test-phases, we used 106 true and 106 false samples. Finally, we used this balanced pool of requirement pairs and randomly shuffled it in order to prepare for training and testing. Each *requirement*-pair defines a sample in the dataset. After this, preprocessing of the data is complete and reasonable *features* can be defined.

9.4.2. Feature Definition

The decision which features to choose is a crucial step for a classifier (Hall, 1999). The used features are *n-grams*, *POS-tags*, and the *direction of the "requires"-dependency*.

- *n-gram features*:
  The extracted tokens/words (i.e., unigrams) as well as the sequences of two to four adjacent tokens were used as features (i.e., bigrams, trigrams, and fourgrams were used). For the remainder of this chapter these *n-grams* are called *n-gram terms*.
  For instance, n-grams of the word sequence "automatic identification of dependencies" would be the following:
  - Unigrams:

    {"automatic", "identification", "of", "dependencies"}
  - Bigrams:

    {"automatic identification", "identification of", "of dependencies"}
  - Trigrams:

    {"automatic identification of", "identification of dependencies"}
  - Fourgram:

    {"automatic identification of dependencies"}
- *POS-tag features*:
  In addition to the n-gram features, POS-tags which were determined during the preprocessing phase were used as additional features. Since the sequence of POS-tags can also be a useful information when detecting a dependency, different sequences of adjacent POS-tags ranging from two to four (i.e., bigrams, trigrams, and fourgrams of POS tag labels) were also used as features.

- *direction of the "requires"-dependency*:
  Since our dataset consists of dependencies of type "requires", the direction of the dependency is an essential information which needs to be considered as a separate feature (see Section 9.4.1). In other words, a "requires"-dependency for the requirement-pair (R_x, R_y) is not the same as a "requires"-dependency for its counterpart-pair (R_y, R_x). Due to this fact, a new binary feature is introduced which compares the two IDs of both requirements of the respective requirement pair. The comparison uses the lower operator to compare both IDs. It returns "true" if the ID of the first requirement is lower than the ID of the second requirement (e.g., $1 < 2$ for the case R_1 *requires* R_2), and "false" otherwise (e.g., $2 < 1$ for the case R_2 *requires* R_1).

In order to adequately learn and predict dependencies between two requirements, *meaningful* information about the two requirements of a requirement-pair must be provided to the classifier. Therefore, the title and the textual description (i.e., detailed information about the requirement) of both requirements were extracted and tokenized as already mentioned before (i.e., n-gram features). For each requirement-pair (R_x, R_y), we applied *term frequency-inverse document frequency* (TF-IDF) (Wu et al., 2008) on each n-gram term of the title and description of R_x and R_y, in order to determine the relevance of the respective n-gram term. The TF-IDF measurement combines *term frequency* (TF) of a term with its *inverse document frequency* (IDF) in order to get the importance of a specific term. For each requirement pair (R_x, R_y), all determined TFIDF values of the pair were combined into a single vector, whereby the TF-IDF value of each n-gram feature of R_x was multiplied with the weight 2 and the TF-IDF value of each n-gram feature of R_y was multiplied with the weight 1. The combination of having a higher weight for the n-gram features of the first requirement and including the direction of the "requires"-dependency feature, allows the classifier to sharply distinguish between a requirement-pair (R_x, R_y) and its counterpart (R_y, R_x). A sequence of all feature values of a sample represents the input vector for this sample.

Formula 9.1 demonstrates the exact setting of a feature vector $\vec{v}$ for a requirement-pair (R_x, R_y), whereby i and j represent one specific element from the complete ngram set of a given requirement.
Furthermore, the function $TFIDF$ calculates the TF-IDF value of a given ngram term and $\vec{POS}(R_x)$ represents the complete vector containing all determined POS tags of the given requirement R_x.

$$\begin{aligned}\vec{v}(R_x,R_y) = &\bigcup_{i\in ngrams(R_x)} \{TFIDF(i).2\} \cup \\ &\bigcup_{j\in ngrams(R_y)} \{TFIDF(j).1\} \cup \\ &\bigcup_{k\in ngrams(R_x)} \vec{POS}(k) \cup \\ &\bigcup_{l\in ngrams(R_y)} \vec{POS}(l) \cup (x<y)\end{aligned} \tag{9.1}$$

9.4.3. Classification

For the *classification* step, the *requirement*-pair dataset is used to learn a model. Thereby, our approach trains a classifier which is then used to determine new requirement dependencies. We trained and tested four different classifiers (*Naive Bayes, Linear Support Vector Classifier, k-Nearest Neighbors Classifier, Random Forest*) from SciKit-Learn library[§] (Pedregosa et al., 2011) in order to compare the prediction

[§]Scikit-learn: `http://scikit-learn.org`

quality of the classifiers. *Naive Bayes* classification is based on a very simple concept which takes the *Bayes Theorem* into account for prediction purposes. The algorithm does not require large training sets and has shown to work efficiently in case of document classification (Metsis et al., 2006). A Linear Support Vector Classifier uses a Support Vector Machine with a linear kernel in order to classify samples in hyper-space. Similar to *Naive Bayes* classifiers, prediction models based on a *Support Vector Machine* have also proven to perform well in case of document classification (Joachims, 1998). *Random Forest* represents a combination of many uncorrelated decision trees. Although single decision trees can run into over-fitting problems quite easily, ensemble classifiers such as *Random Forests* very unlikely tend to encounter such issues (Breiman, 2001). Finally, *k-Nearest Neighbors* classification (KNN) is a very basic approach that predicts the class based on the majority class of the k closest (i.e., lowest distance) neighbors of a sample. In general, KNN has shown to perform well regarding document classification, however, the classifier's prediction quality heavily depends on the quality of the words being used (Trstenjak et al., 2014).

9.4.4. Feature Extraction and Feature Selection with Grid Search

For the learning purpose, we support each classifier with several features (see Table 9.2) such as:

- *vectorizer max features* determines the number of used features. A value of 300 means that the 300 most frequent features in the training set are used for training and *none* means that there is no limitation and the system can use all features.
- *vectorizer max df* is for ignoring some specific words which can occur very often in the text such as "and", "or", .., etc. A value of 0.5 means that the classifier should ignore all the words which have an overall occurrence rate that is higher than 50%.
- *vectorizer min df* indicates that the vectorizer should ignore words which generally occur very rarely. A value of 3 means ignoring all the words which occur less often than three times.
- *vectorizer n-gram range* is for the determining of n-grams. For instance, (1,2) indicates the analysis of uni- and bigrams.

Grid Search TF-IDF features	
Features	possible values
vectorizer max features	{none, 300, 400, 500, 1000}
vectorizer max df	{0.5, 0.7, 0.9, 1.0}
vectorizer min df	{1, 2, 3}
vectorizer n-gram range	{(1, 1), (1, 2), (1, 3), (1, 4)}

Table 9.2.: Possible parameter combinations for the TF-IDF Vectorizer

In order to extract and select suitable features for the given classifiers, we provide a predefined set of possible parameter combinations for the TF-IDF vectorizer (see Table 9.2). Thereby, *Grid Search* in combination with k-fold *Cross-validation* (Stone, 1974) (with $k = 3$) is applied in order to find a suitable parameter combination for the vectorizer which then extracts and selects appropriate features for the respective classifier. Using *Grid search* with 3-fold *Cross Validation*, the dataset is divided into three equally sized folds whereby two folds are used to train the classifier with the respective parameter combination and one fold to evaluate the prediction quality. To evaluate the predication quality, the F1 measure is used. This step is

repeated for each possible parameter combination from the given parameter set. Thereafter, the parameter combination which achieves the highest F_1 value for the given classifier is chosen.

9.5. Evaluation

As already mentioned in Section 9.4.4, *k-fold Cross Validation* was used to evaluate the classifiers. The dataset was split into three (i.e., $k = 3$) equally sized parts. Two folds (i.e., training dataset) were used to train the classifier and one fold (i.e., test dataset) to evaluate the prediction quality of the trained classifier model with unseen data. In total, there exist $k = 3$ rounds. This procedure was then repeated with a different splitting until all combinations were analyzed, i.e., all three rounds were done. Thereafter, the average result of all rounds for each measure was computed. The samples of the test dataset are new requirement-pairs that have not been used for training (i.e., are not part of the training set). This way, the classifier predicts for each tested *requirement-pair* whether its first requirement (called R_x) requires its second requirement (called R_y) or not. More formally, the classifier answers the question $R_x \rightarrow R_y$ for each *requirement-pair* in the test set. The predicted class (i.e., $true$ or $false$) can then be compared with the actual class (i.e., value of the *existence label*) of the respective *requirement-pair*. Based on this comparison, the overall prediction quality is measured by using the *Precision*, *Recall*, and *F1* measures (see Formulae 9.2- 9.4). The *precision* presented in Formula 9.2 is the fraction of relevant requirement dependencies *tp* (i.e., true positives) among all the retrieved requirement dependencies consisting *tp* and *fp* (i.e., false positives). Formula 9.3 indicates the *recall* which is the fraction of relevant requirement dependencies *tp* that have been retrieved over the total amount of relevant requirement dependencies *tp* and *fn* (i.e., false negatives). Moreover, F_1 in Formula 9.4 indicates the *harmonic mean* which is based on the calculated *precision* and *recall* values. In order to individually compare the prediction quality of both classes (i.e., $true$ and $false$), these measures are applied separately for each of both classes. Additionally, the *Area under the Receiver Operating Characteristic Curve* (AUROC) (DeLong et al., 1988) is calculated. AUROC defines the area between the ROC curve and the horizontal x-axis. More formally, AUROC represents the integral of the ROC curve. The resulting value is then a characteristic value of how imbalanced the prediction quality for different classes is. This way, a possible prediction imbalance for a certain class (i.e., the classifier can predict one class very well but the prediction quality of the other class is very low) can be easily observed from the evaluation results.

$$precision = \frac{tp}{tp + fp} \tag{9.2}$$

$$recall = \frac{tp}{tp + fn} \tag{9.3}$$

$$F_1 = 2. \frac{precision \cdot recall}{precision + recall} \tag{9.4}$$

As already mentioned in Section 9.4.3, four different classifiers were evaluated. Training and testing are both performed by using the features described in Section 9.4.2. In order to check the suitability of POS-tags as separate features, each classifier was evaluated twice (one time without POS-tags and another time by including POS-tags as additional features).

	Dependency exists (true)				No dependency exists (false)			
Classifier	Precision	Recall	F1	AUROC	Precision	Recall	F1	AUROC
Naive Bayes	0.944	0.654	0.773	0.787	0.640	0.941	0.762	0.213
Linear SVM	0.947	0.692	0.800	0.828	0.667	0.941	0.780	0.172
k-NN	0.842	0.615	0.711	0.767	0.583	0.824	0.683	0.233
Random Forest	0.950	0.731	0.826	0.865	0.696	0.941	0.800	0.135

Table 9.3.: Precision, Recall, and F1 Scores for the different Classifiers without POS Tags as Features

	Average		
Classifier	Precision	Recall	F1
Naive Bayes	0.824	0.767	0.768
Linear SVM	0.836	0.791	0.792
k-NN	0.740	0.698	0.700
Random Forest	0.849	0.814	**0.816**

Table 9.4.: Average Precision, Recall, and F1 Scores for the different Classifiers without POS Tags as Features

Table 9.3 and 9.4 show the evaluation results of the first evaluation scenario. In the first evaluation scenario, POS-tags are not included in the set of used features. Hence, the used features are TF-IDF values of n-grams and the *direction* feature (see Section 9.4.2). Table 9.3 and 9.4 give an overview of the measured values (*Preion*, *Recall*, *F1*, and *AUROC*) achieved by each classifier for the $true$ class (i.e., *requires*-dependency exists), the $false$ class (i.e., no *requires*-dependency exists), and the weighted average of both classes in terms of *Precision*, *Recall*, and *F1*. The values for *Precision*, *Recall*, *F1*, and *AUROC* can range from 0.0 to 1.0. High values of *Precision*, *Recall*, and *F1* indicate a high prediction quality, whereas the *absolute* distance between the *AUROC* value and the number 0.5 (which corresponds to the area of a random predictor (Butkiewicz et al., 2009)) should be as large as possible. The results in Table 9.3 and 9.4 show that all four classifiers can achieve a good prediction quality for both classes (i.e., $true$ and $false$) when not taking POS-tags into account. This means that the existence of a *requires*-dependency can be predicted very well. In particular, the *Linear SVM* and *Random Forest* classifiers show a high prediction rate for both classes. Hence, the average values for both classes shown in Table 9.3 and 9.4 in terms of precision (Linear SVM: 0.836, Random Forest: 0.849), recall (Linear SVM: 0.791, Random Forest: 0.814) and F1 (Linear SVM: 0.792, Random Forest: 0.816) are also remarkable. Moreover, the *Naive Bayes* approach also achieves decent results for the given setting. Although the Naive Bayes approach assumes that all features of the feature vector are independent (which is usually wrong for natural language text), the algorithm has proven to also work well with dependent features extracted from text (Čubranić, 2004) (see Section 9.4.3). Furthermore, *k-Nearest Neighbors* classifier cannot predict both classes as well as the other classifiers when using only n-grams and the dependency-direction as features. By taking a look at the AUROC values of both classes for each estimator, a good prediction balance for both classes can also be noticed. Thereby, *Random Forest* achieves very notable results (0.865 for $true$ class and 0.135 for the $false$ class).

The evaluated results of the prediction quality for the second evaluation scenario are presented in Table 9.5 and Table 9.6. In this scenario, POS-tags are introduced as additional features and included in the set

of used features. In comparison to the F1 values achieved in the previous evaluation step, only the *k-Nearest Neighbors* classifier shows better prediction quality in terms of the F1 measure for both classes and consequently also on average (0.793). Additionally, this estimator also achieves the best prediction quality of all classifiers for this specific evaluation scenario. The AUROC values of both classes for this classifier confirm the existence of a good prediction balance for both classes (0.801 for $true$ class and 0.199 for the $false$ class). However, *Random Forest* shows the best prediction balance (AUROC of $true$ class: 0.853, AUROC of $false$ class: 0.147). The inclusion of POS-tags as additional features significantly lowers the prediction quality of all other three classifiers. Most notably, the highest loss of the prediction rate (when compared to the first evaluation scenario) can be noticed for the *Naive Bayes* classifier. On average, this estimator can only achieve a precision of 0.725, a recall of 0.674, and a F1 value of 0.676. Furthermore, the AUROC values of this classifier for both classes (0.756 for positive and 0.244 for negative lie closer to the area of 0.5) which indicates a prediction imbalance issue. Due to the nature of the Naive Bayes approach, it can determine the probability of a given word for a class quite well (Metsis et al., 2006). However, the variety of different words overwhelms the variety of different POS-tags (i.e., each POS-tag represents a large set of many different words). Consequently, the probability that a specific class contains a certain POS-tag (e.g., "verb") more likely might not be an appropriate basis for making a specific decision to select this certain class.

	Dependency exists (true)				No dependency exists (false)			
Classifier	Precision	Recall	F1	AUROC	Precision	Recall	F1	AUROC
Naive Bayes	0.833	0.577	0.682	0.756	0.560	0.824	0.667	0.244
Linear SVM	1.000	0.577	0.732	0.792	0.607	1.000	0.756	0.208
k-NN	0.870	0.769	0.816	0.801	0.700	0.824	0.757	0.199
Random Forest	0.833	0.769	0.800	0.853	0.684	0.765	0.722	0.147

Table 9.5.: Precision, Recall, and F1 Scores for the different Classifiers including POS Tags as Features

	Average		
Classifier	Precision	Recall	F1
Naive Bayes	0.725	0.674	0.676
Linear SVM	0.845	0.744	0.741
k-NN	0.803	0.791	**0.793**
Random Forest	0.774	0.767	0.769

Table 9.6.: Average Precision, Recall, and F1 Scores for the different Classifiers including POS Tags as Features

Overall, the prediction quality of all classifiers is decently high. In other words, all classifiers are able to identify correct dependencies by just providing them all possible pairs of requirements for which the *requires*-dependency should be validated. Finally, it is important to mention that it is not sufficient if the classifier only predicts the existence of a dependency between two requirements of a given requirement-pair correctly. This is due to the reason that the direction of the *requires*-dependency must also be exact in order to be correct. In other words, for example, assuming that R_1 requires R_2 and R_2 does *not* require R_1, a predicted *requires*-dependency for the pair (R_1, R_2) would be considered as correctly classified example. However, a *requires*-dependency for the pair (R_2, R_1) would be considered as incorrectly classified exam-

ple. Taking this strictness aspect into account when taking a look at the results, all the results achieved by the classifiers are even more remarkable.

9.6. Threats to Validity

This section describes the threats to internal and external validity of our approach.

9.6.1. Internal Validity

From our point of view there are some threats to the internal validity of our automated dependency detection approach. One threat of our approach is the validation of our results with a small dataset. Our results are based on data collected from 182 participants which is insufficient for a correct validation. There still exists the necessity to train and to test our approach with further datasets in order to prove the results presented in Tables 9.3 - 9.6. Furthermore, suitable n-gram features were extracted and selected based on a limited set of reasonable parameters by using Grid search (see Table 9.2). A more appropriate and intelligent strategy for feature extraction and selection would find different features which may achieve even higher prediction quality.

As described in Section 9.4.1, we shuffled our dataset consisting of requirement-pairs and applied a random undersampling strategy in order to counteract class imbalance issues. However, undersampling leads to loss of important information which could be useful for the classifier in order to further improve its prediction quality. Furthermore, a different randomization of the dataset could also lead to slightly different results compared to the values given in the Tables 9.3 - 9.6. Additionally, in the case of the cold start problem, the performance of the classifiers to predict the correct dependency between two requirements of a given requirement-pair could be inadequate.

9.6.2. External Validity

For the analysis of our approach in training- and test-phases, we used a dataset collected in a user study with 182 participants. Within the scope of future work, we will test our system with different datasets in order to confirm our results. Testing our approach with different datasets may lead to small deviations of the values presented in Tables 9.3 - 9.6. Due to the experimental testing of our approach, a test in the live-operation may also lead to different results. Furthermore, our dataset consists of requirements described in German and the system is optimized to perform well with text provided in German language. Analyzing a dataset based on a *different* language can lead to different evaluation results.

9.7. Conclusion and Future Work

In this chapter we introduced an intelligent approach for the automated identification of requirement dependencies of type *requires*. Our approach analyzes the title and the description of each requirement and detects dependencies by using NLP techniques. The data for the dataset was collected in a user study. Before the training of the system, *preprocessing* techniques were applied in order to prepare a dataset for the classifiers and to filter irrelevant information. Afterwards, we analyzed our approach with different classifiers such as *Naive Bayes, Linear SVM, k-Nearest Neighbors,* and *Random Forest* using several features determined via Grid search in order to identify the most suitable feature combinations for each

classifier. The results show that average values of *precision, recall,* and F_1 without POS-Tags generally perform better than including POS-Tags for all applied classifiers except the *k-Nearest Neighbors* classifier. Furthermore, a *Random Forest* classifier without POS-Tags returns the best F_1 value on average which correctly predicts the dependencies between requirements with a probability of $\sim$82%.

Within the scope of future work, we plan to introduce an additional preprocessing step where we will replace words with their synonyms and combine similar words by using an *online thesaurus*. Moreover, the currently used random undersampling strategy was necessary to avoid undesired class imbalance issues (see Section 9.6.1). However, it could be replaced with more advanced and sophisticated sampling techniques in order to further improve the overall prediction quality of the system (Bowyer et al., 2011; Galar et al., 2012). Furthermore, we found out that many of the dependencies which have not been found often by the study participants are those which cannot be observed directly from the title and description of the requirements. In order to detect such dependencies, further domain knowledge is required. Although we did not do any optimizations regarding this aspect, our classification models were able to find some of such dependencies. This shows that the classifiers are already able to generalize in such a way that they can discover and consider some latent patterns. However, further improvements have to be made in order to improve the prediction quality for such dependencies. Finally, our developed approach can be used as a recommender system (e.g., in terms of a hybrid solution of different prediction models) which identifies dependencies between a set of requirements and recommends them to requirement engineers.

Chapter 10

ASP-based Knowledge Representations for IoT Configuration Scenarios

Parts of the contents of this chapter have been published in (Felfernig et al., 2017b). The author of this thesis provided major parts of this chapter in terms of writing, literature research, knowledge base definition, and implementation.

10.1. Abstract

The purpose of this chapter is to introduce basic application scenarios for configuration technologies in Internet of Things (IoT) product domains. We show how to represent configuration knowledge in the domain of *smart homes* on the basis of Answer Set Programming. In this context, we introduce different configuration model elements and constraint types and show their corresponding Answer Set Programming representation in a way that is also useful for beginners. We conclude the chapter with a discussion of open issues for future work.

10.2. Introduction

Configuration is one of the most successfully applied AI technologies (Stumptner, 1997; Felfernig et al., 2014a). It is a specific type of design activity where a product is configured on the basis of a set of already defined component types and corresponding constraints that restrict the way in which component instances can be combined. A *configuration task* is defined in terms of a generic product structure, a corresponding set of constraints, and a set of requirements (often also denoted as customer requirements) that additionally restrict the set of possible solutions. A solution (configuration) for a configuration task is represented by a set of component instances, their connections and attribute settings which altogether are consistent with the constraints and requirements included in the configuration task definition.

There is a multitude of application domains for knowledge-based configuration – example domains are the automotive sector, financial services, operating systems, software product lines, and railway interlocking systems (Felfernig et al., 2014a). Configuration technologies nowadays become increasingly popular in different kinds of Internet of Things (IoT) (Atzori et al., 2010) scenarios. The Internet of Things is an emerging paradigm that envisions a networked infrastructure which enables the interconnection of

devices (things) anyplace and anytime. In the IoT context, configurators can be applied, for example, to the identification of ramp-up configurations (self-configuration), i.e., to figure out which components (potential software and hardware) are needed in a certain IoT setting. Configuration technologies can also be applied during runtime where, for example, a configurator helps to identify paretooptimal configurations of communication protocols with regard to criteria such as *performance* and *cost of data transfer*.

The size and complexity of configuration problems in the IoT domain often does not allow the application of basic configuration knowledge representation and reasoning such as constraint satisfaction (Tsang, 1993). Smart homes often consist of hundreds or even thousands of different components and constraints – such scenarios are in the need of a component-oriented knowledge representation that is easy to use and maintain (Felfernig et al., 2014a; Hotz et al., 2014). Open source constraint-based approaches do not support such a representation and existing component-oriented commercial environments are based on proprietary knowledge representations with limitations also in terms of standardization. An alternative to constraint-based knowledge representations especially useful for large and complex configuration domains is Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988; Soininen and Niemelä, 1998). ASP supports the definition of component hierarchies and related constraints in a declarative way (which is not possible with basic CSP-based configuration environments). Potential component instances have to be pre-defined, i.e., in its basic form ASP does not support pure component generation during runtime.

There exist a couple of research contributions related to the application of answer set programming in the configuration context. Soininen and Niemelä (Soininen and Niemelä, 1998) can be considered as pioneers who first showed the application of ASP to represent and solve configuration tasks. A resulting configuration environment is presented, for example, in (Tiihonen et al., 2003; Felfernig et al., 2014a). An object-oriented layer to answer set programs has been introduced by (Falkner et al., 2015). In this work, configuration tasks can be represented on an object-oriented level without being forced to take into account specific details of ASP-based configuration knowledge representations. Thus, this work can be seen as a contribution to improve the applicability of ASP technologies especially in terms of reducing efforts related to knowledge base development and maintenance. Feature model related ASP representations are introduced in (Myllärniemi et al., 2014). An approach to the testing of object-oriented models on the basis of ASP is introduced in (Falkner et al., 2012); in this context it is shown how UML-based configuration knowledge representations can be represented in ASP and how positive and negative test cases can be represented and included for the purpose of supporting unit tests on knowledge bases. Friedrich et al. (Friedrich et al., 2011) introduce an approach to re-configuration in ASP – in this context, a reconfiguration can be considered as a set of changes to an already existing configuration such that new requirements are taken into account. Finally, Teppan et al. (Teppan and Friedrich, 2016) introduce a hybrid approach that integrates constraint solving with ASP. A major advantage of this integration is that the grounding bottleneck* in answer set programming can be transformed into a more efficiently solvable search problem in constraint programming.† The major focus of this chapter are ASP-based knowledge representations. For an overview of different further approaches to configuration knowledge representation we refer to (Felfernig et al., 2014a).

*See Section 10.5.

†Although CSPs often do not support flexible (component-oriented) knowledge representations, they have the potential to support ASP-based reasoning processes (e.g., in terms of increasing efficiency).

The contributions of this chapter are the following.[‡] First, we introduce ASP-based configuration knowledge representations in the context of Internet Of Things (IoT) scenarios. Second, our aim is to provide easy to understand examples (for ASP newbies) of how to represent configuration knowledge in ASP and also to show limitations of ASP knowledge representations. Third, we discuss different issues for future research that will help to accelerate a broad application of ASP technologies in knowledge-based configuration.
The remainder of this chapter is organized as follows. In Section 10.3 we discuss IoT-related configuration domains and introduce a smart home configuration model which is used as working example throughout the chapter. In Section 10.4 we show how to translate individual model elements into a corresponding ASP-based representation. In this context we also sketch how ASP solvers operate to determine a solution for a configuration task (Section 10.5). In Section 10.6 we sketch the role of ASP solving in our IoT-related research project. The chapter is concluded with a discussion of open research issues (Section 10.7).

10.3. IoT Domains and Configuration Models

In the following we provide an overview of IoT domains where the application of ASP-based configuration technologies is reasonable. In this context, we introduce a simplified configuration model from the domain of smart homes in order to show different facets of ASP-based configuration knowledge representations.
Air Pollution Monitoring. Air pollution monitoring systems help to ensure healthy living conditions, for example, in cities. An issue in this context is the distribution of sensors in a city topology that assures a representative collection of measurement data. This data is analyzed on the basis of different types of learning algorithms that help to figure out in which contexts which actions have to be triggered. Examples of related actions are a general warning to leave the house, reduced speed limits on highways, recommendations to groups (e.g., schools) in terms of the maximum time that should be spent outdoors, and warnings regarding the malfunctioning of filter equipments in industrial production. In air pollution monitoring, configuration technologies can be used to select the type and placement of sensors given a specific topology (e.g., a topology of a city) and also to select the types of algorithms that should be used for data analysis in certain contexts.

Health Monitoring. Health monitoring solutions can be based on different types of data that can be used to determine recommendations related to factors such as eating behavior, sports activities, sleeping times, and also data about the body condition. Many tools already allow the manual entering of consumed food, however, in future scenarios such information will be available on the basis of standardized data exchange protocols. Measurement of sports activities and sleeping time is already included in many commercial solutions. Finally, detailed information about the physical condition of a person is not taken into account in many of the existing tools. In such scenarios, configuration technologies can be used to parametrize the underlying algorithms, for example, recommended heart rates when doing physical practices depend on the age, gender, and weight of a person (and further physical parameters). Whether specific food items can be recommended or not depends, for example, on potential allergies of a person. Finally, especially in group sports (e.g. football or tennis), the type of training also depends on the participating persons. For example, if three persons are participating in a tennis training session and one person has a bad physical condition, this has an impact on the selection of exercise units for this group.

[‡]The work presented in this chapter has partially been conducted within the scope of the Horizon 2020 Project AGILE (Adoptive Gateways for dIverse MuLtiple Environments, 2016–2018).

Energy Production and Management. Energy production is in the need of configuration technologies in various scenarios, for example, wind turbines must be configured and parametrized in order to be able to maximize energy production in a certain environment. Knowledge about where and when a higher amount of energy will be needed can trigger a corresponding reconfiguration of the load factor of water reservoirs. In the context of private energy production, configuration technologies can help to rearrange energy consumption times, for example, when to recharge the electric car or when to activate the washing machine. Especially in the context of energy management in buildings, reconfiguration technologies can play a role by supporting the change of building parameters depending on given environmental data such as temperature, weather conditions, and forecasts.

Enhanced Retail Services. In-store shopping is based on specific distributions of sensors and other devices such as information displays. During the ramp-up phase of such an application it has to be assured that customer location sensors are distributed in a reasonable fashion and information displays are positioned in such a way that the information can be easily seen by customers. In such scenarios, configuration technologies can be applied in order to determine the amount of sensors needed, the positioning of information displays, and also to determine the layout of the whole shop depending on the product assortment that should be offered to a customer.

Animal Monitoring. There exist a couple of scenarios where information about animal locations and information about the physical condition of animals is important. For example, in wildlife scenarios where animals are spread over huge and not accessible areas, it is important to provide an infrastructure for animal monitoring that is not based on physical presence of human administrators. In such contexts, for example, different types of drones can be used to support data collection. Depending on the region size and topography and requirements regarding the amount of data to be collected, drones have to be configured in a way that optimizes the trade-off between energy consumption, range, and support of the defined data collection requirements. In such scenarios, configuration technologies can be useful to support the complete configuration of the needed data collection equipment.

Smart Homes. A simplified smart home configuration model is depicted in Figure 10.1. Smart homes (Leitner et al., 2016) include functionalities for actively supporting persons in their daily life. Examples thereof are *intelligent light management* that allows to (semi-automatically) adapt the illumination of rooms depending on the time of the day and season, *energy management* that supports intelligent air conditioning for whole buildings, *security management* (e.g., when nobody is at home), and functionalities related to the support of *ambient assisted living* scenarios with related functionalities such as automated fall detection. In such scenarios, configuration technologies can be used to design in detail which smart home hardware and software components (e.g., sensors, actuators, and apps) have to be installed in which part of the building. The smart home model depicted in Figure 10.1 will serve as working example in this chapter to demonstrate ASP-based configuration knowledge representations.

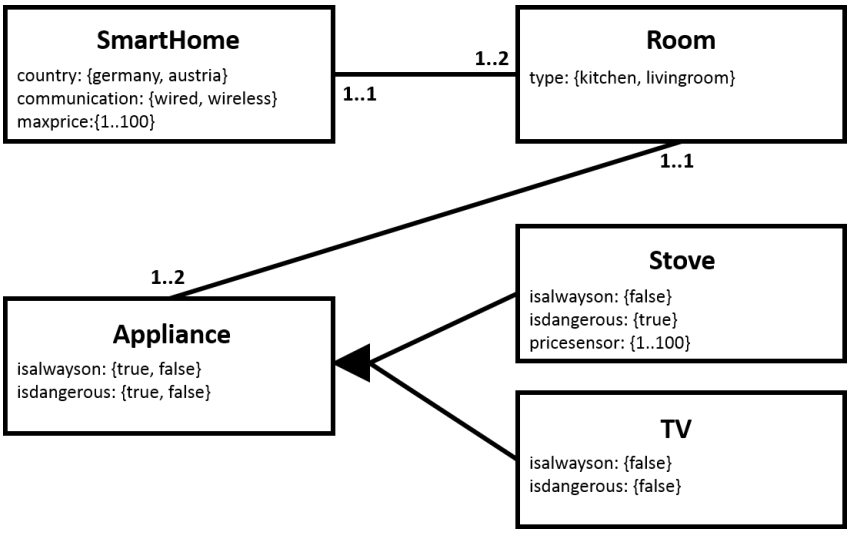

Figure 10.1.: Simplified configuration model (reduced #component types, #attributes, domains, and multiplicities) of a *smart home* used for demonstration purposes. Additional constraints are presented in our discussion of ASP-based configuration knowledge representations.

10.4. Configuration Knowledge Representation in ASP

In the following we will show how to represent configuration knowledge for a simplified configuration model from the domain of smarthomes (see Figure 10.1).[§]

(a) *Potential Component Instances*. In the ASP context, all decision variables have to be pre-specified. In our example model shown in Figure 10.1, three different component types are included which are represented by the predicate names *smarthome*, *room*, and *appliance* (with the subtypes *stove* and *room*). For these component types we have to specify the maximum amount of instances that can be part of a related smarthome configuration (see Figure 10.2), i.e., one *smarthome* (e.g., *psmarthome*(1) denotes a potential instance of *smarthome* with id 1), two instances of *room*, and 4 instances of each subtype of *appliance*. The integer number arguments (e.g., *proom(2;3)*) in the ASP facts serves as a unique key to distinguish the (potential) instances.

```
psmarthome(1).
proom(2;3).
pstove(4;5;6;7).
ptv(8;9;10;11).
```

Figure 10.2.: Definition of potential smart home component instances using Answer Set Programming (ASP) notation. For example, *proom(2;3)* denotes two potential instances of type *room*.

(b) *Component Types and Instances*. For a configuration, it has to be decided which of the potential instances shall be included. Therefore we establish an association of the aforementioned definitions of potential instances with the ”real” component instances that can be included in a configuration (see Figure 10.3).

[§]For demonstration purposes we use the syntax of the *clingo* environment (see potassco.sourceforge.net).

For each potential instance it has to be decided whether to include this as an instance in a configuration or not, for example in our configuration knowledge base, each potential room instance can be part of a configuration or not (this is specified by the lower and upper bounds of the corresponding rule).

```
0{ smarthome(X) }1 :- psmarthome(X).
0{ room(X) }1 :- proom(X).
0{ stove(X) }1 :- pstove(X).
0{ tv(X) }1 :- ptv(X).
```

Figure 10.3.: Definition of component types (in ASP). For example, each potential instance of type room (i.e., *proom(X)*) can be part of a configuration as *room(X)*.

(c) *Generalization Hierarchies*. Generalization hierarchies allow to further categorize different component types (see Figure 10.4). For demonstration purposes, we represent *stove* and *tv* as (disjunctive) subtypes of the component type *appliance* – an alternative to introducing a *type* attribute similar as for *room*. We also include a rule that assures that instances of appliances are instantiated to *stove* or *tv*. Note that disjunctiveness between subtypes is assured by definition (see Figures 10.2 and 10.3).
As we are not interested in incomplete configurations (e.g., instances of *appliance* that are not refined to *stove* or *tv*), we add a constraint that ensures that each *appliance* instance is either refined to a *stove* instance or a *tv* instance. This is only necessary when users are allowed, for example, to include component instances represented by corresponding facts (e.g., *appliance(4)*).

```
appliance(X) :- stove(X).
appliance(X) :- tv(X).
:- appliance(X), not stove(X), not tv(X).
```

Figure 10.4.: Defining generalization hierarchies (in ASP). For example, each *stove* is an *appliance* and each *tv* is an *appliance*, and vice-versa, each *appliance* is either a *stove* or a *tv*.

(d) *Attributes*. For the defined component type attributes, we have to introduce attribute domain definitions (see Figure 10.5).

```
dommaxprice(1..100).
domcountry(germany;austria).
domcommunication(wired;wireless).
domtype(kitchen;livingroom).
domisalwayson(true;false).
domisdangerous(true;false).
dompriceofsensor(60).
```

Figure 10.5.: Attribute domain definitions (in ASP). For example, *dommaxprice* represents an (integer) attribute that will be used to specify the maximum price of a *smarthome* solution. For simplicty, we only include price information related to *stoves* – see also Figure 10.6.

Attribute domain definitions have to be associated with the corresponding component types, for example, the domain definition *domcountry* of the attribute *country* has to be associated with the component type

smarthome (see Figure 10.6). With ASP choice rules we enforce that each instance has exactly one domain value for each of its attributes. Generalization is covered in a natural way: see the right-hand-side of the last two rules in the figure. As attributes are created only for existing instances (but not for potential instances), spurious solutions (such as arbitrary attribute settings for unused potential instances and their combination) are avoided.

```
1{country(X,Y):domcountry(Y)}1:- smarthome(X).
1{communication(X,Y):domcommunication(Y)}1:- smarthome(X).
1{maxprice(X,Y):dommaxprice(Y)}1:-smarthome(X).
1{type(X,Y):domtype(Y) }1:- room(X).
1{isalwayson(X,Y):domisalwayson(Y)}1:- appliance(X).
1{isdangerous(X,Y):domisdangerous(Y)}1:- appliance(X).
1{priceofsensor(X,Y):dompriceofsensor(Y)}1:- stove(X).
```

Figure 10.6.: Associating attributes with component types (in ASP). For example, *country* is an attribute associated with *smarthomes*. On an instance level, attribute instances are only generated if corresponding component instances exist, i.e., attribute instances are only created when necessary.

If domain definitions are reduced in subcomponent types, this can be expressed in a corresponding ASP rule, for example, *isdangerous(X,false):- tv(X).* expresses the fact that a *tv* set is not considered as a dangerous appliance (see Figure 10.7).

```
isalwayson(X,false) :- stove(X).
isdangerous(X,true) :- stove(X).
isalwayson(X,false) :- tv(X).
isdangerous(X,false) :- tv(X).
```

Figure 10.7.: Reducing ASP attribute domain definitions, for example, in generalization hierarchies. For example, the *isdangerous* attribute of type *appliance* is reduced to *true* if the corresponding component is a *stove*.

(e) *Associations and Multiplicities*. Associations between component types on the model level are represented in terms of binary predicates on the ASP level (see Figure 10.8). We use ASP rules to define potential links, e.g. *smarthomeroom*, with the allowed minimum and maximum multiplicities of the association.

```
1{ smarthomeroom(X,Y): room(Y) }2 :- smarthome(X).
1{ smarthomeroom(Y,X): smarthome(Y) }1 :- room(X).
1{ roomappliance(X,Y): appliance(Y) }2 :- room(X).
1{ roomappliance(Y,X): room(Y) }1 :- appliance(X).
```

Figure 10.8.: Defining associations and corresponding multiplicities (in ASP). For example, each *smarthome* has 1-2 associated components of type *room*.

(f) *Incompatibility Constraints*. Such constraints typically specify incompatibilities regarding the combination of specific component types or simply combinations of incompatible attribute values. An example of an incompatibility is represented by the following constraint that expresses the fact that a *tv* should not be situated in a room of type *kitchen* (see Figure 10.9).

```
:- roomappliance(X,Y), type(X,kitchen), tv(Y).
```

Figure 10.9.: Defining incompatibility constraints (in ASP). For example, no *tv* should exist in a *kitchen*.

(g) *Requires Constraints*. Requirements relationships describe situations where the integration of a certain component or the selection of a certain attribute value also requires the integration/selection of further component types/attribute values. An example constraint is the following: *smarthomes* in *Austria* must have two rooms (the corresponding ASP representation is shown in Figure 10.10).

```
2{ smarthomeroom(X,Y): room(Y) }2 :-
 smarthome(X), country(X,austria).
```

Figure 10.10.: Defining requires constraints (in ASP). For example, *smarthomes* in *Austria* include at least two *rooms*.

(h) *Resource Constraints*. Resource constraints specify producer and consumer relationships, for example, a resource (*producer*) could be the money available for the smarthome (*maxprice* specified by the customer) and the consumers could be the installed sensors (represented by the attribute *priceofsensor*). A corresponding resource constraint could indicate that the sum of the prices of all sensors must not exceed the upper price limit specified by the customer (attribute *maxprice*). An implementation of a resource constraint in ASP is shown in Figure 10.11.

```
sensorprice(T) :- T = #sum{ PR, A : priceofsensor(A, PR)}.
:- smarthome(X), maxprice(X,Y), sensorprice(P), P > Y.
```

Figure 10.11.: Defining resource constraints (in ASP). For example, the price of a *smarthome* (represented by *sensorprice* only) must not exceed the *maxprice* defined by the customer.

(i) *Navigation Constraints*. ASP allows to represent complex constraints which require navigation between instances. For example, a stove in one room excludes further stoves in other rooms of the same smarthome (see Figure 10.12). This can also be interpreted as a further example of an incompatibility constraint (see Figure 10.9). Even recursively defined predicates can be used in such constraints such that transitive closures (e.g. reachability in graphs) and constraints on them can be expressed. This is a modeling advantage compared to standard constraint solvers.

```
:- stove(A1), roomappliance(R1,A1), smarthomeroom(H,R1),
 R1 != R2,
 smarthomeroom(H,R2), roomappliance(R2,A2), stove(A2).
```

Figure 10.12.: Defining navigation constraints (in ASP). For example, two different *rooms* with a *stove* must not be part of the same *smarthome* configuration. In this context, R1 != R2 assures that two different *rooms* are analyzed with regard to the inclusion of a *stove*.

(j) *Example Customer Requirements*. Having defined the whole configuration knowledge base, customers can specify their requirements with regard to a corresponding smarthome configuration as facts and even more generally as constraints (see Figure 10.13). Examples of such customer requirements are: *the smarthome installation will be located in Austria* and *no dangerous appliances should be installed* (see the following constraints).

```
smarthome(1).
country(1,austria).
:- appliance(X), isdangerous(X,true).
```

Figure 10.13.: Defining requirements (in ASP). For example, the *smarthome* should be in *austria* and no dangerous *appliances* should be included.

10.5. ASP Solving and Limitations

Classical ASP solvers (Gebser et al., 2012) work in two steps: (1) *grounding* which translates the ASP program to a variable-free format and (2) propositional (SAT) *solving* of the grounded program. Figure 10.14 shows a reduced version of our *smarthome* configuration knowledge base.

The knowledge base in Figure 10.14 is a subset of the class diagram in Figure 10.1. It includes a component type *smarthome* with the attribute *country* and a component type *room* with the attribute *type*. A *smarthome* can have 1 or 2 *rooms* but Austrian *smarthomes* must have two rooms (this is defined in terms of an additional constraint). One potential instance is defined for *smarthome* and three potential instances are defined for *room*. The requirements specify the inclusion of a *room of type kitchen*.

The *grounding* results are shown in Figure 10.15 – the relationship to the knowledge base of Figure 10.14 is explained in terms of comments. *ASP facts* of the original knowledge base are also represented as facts in the grounded knowledge base. Variables in rules are removed and the rules are duplicated accordingly, for example, three rules are generated for the three possible room instances: i.e., *proom(X)* in the second rule for generation of instances is replaced with each of the three facts for potential instances and the head of the rule is replaced accordingly. Furthermore, the *count* aggregate is represented in its standard form instead of just curly brackets.

```
% potential instances
psmarthome(1). proom(2;3;4).

% attribute domain definitions
domcountry(germany;austria). domtype(kitchen;livingroom).

% definition / generation of instances
0{ smarthome(X) }1 :- psmarthome(X).
0{ room(X) }1 :- proom(X).

% associating attributes with component types
1{ country(X,Y): domcountry(Y) }1 :- smarthome(X).
1{ type(X,Y): domtype(Y) }1 :- room(X).

% definition / generation of associations
1{ smarthomeroom(X,Y): room(Y) }2 :- smarthome(X).
1{ smarthomeroom(Y,X): smarthome(Y) }1 :- room(X).

% further constraints
:- smarthome(X), country(X,austria),
 not 2{ smarthomeroom(X,Y): room(Y) }2.

% customer requirements
room(2). type(2,kitchen).
```

Figure 10.14.: Restricted version of example *smarthome* configuration model.

In general, grounding can lead to extremely large knowledge bases, especially if rules or constraints entail many variables with a large domain. The domain sizes are multiplied which leads to exponential growth in the number of variables in the worst case. This shows one of the weaknesses of answer set programs - they are not well-suited for problems with *large integer domains or floating point numbers*. Ways to deal with this issue is to combine answer set programming with constraint solving techniques (see, e.g., (Gebser et al., 2015; Teppan and Friedrich, 2016)) and lazy grounding (see, e.g., (Eiter et al., 2018)).

Solving an ASP results in answer sets (solutions). Each solution must be well-founded (i.e., derived from the given facts) and consistent (i.e. not violating any constraint). Figure 10.16 shows all solutions (answer sets) derived from the example in Figure 10.14.

```
% potential instances
psmarthome(1). proom(2). proom(3). proom(4).

% attribute domain definitions
domcountry(germany). domcountry(austria).
domtype(kitchen). domtype(livingroom).

% definition / generation of instances
0<=#count{1,0,smarthome(1):smarthome(1)}<=1.
0<=#count{1,0,room(2):room(2)}<=1.
0<=#count{1,0,room(3):room(3)}<=1.
0<=#count{1,0,room(4):room(4)}<=1.

% associating attributes with component types
1<=#count{1,0,country(1,germany):country(1,germany);
 1,0,country(1,austria):country(1,austria)
 }<=1:- smarthome(1).
1<=#count{1,0,type(2,kitchen):type(2,kitchen);
 1,0,type(2,livingroom):type(2,livingroom)
 }<=1:- room(2).
1<=#count{1,0,type(3,kitchen):type(3,kitchen);
 1,0,type(3,livingroom):type(3,livingroom)
 }<=1:-room(3).
1<=#count{1,0,type(4,kitchen):type(4,kitchen);
 1,0,type(4,livingroom):type(4,livingroom)
 }<=1:-room(4).

% definition / generation of associations
1<=#count{
 1,0,smarthomeroom(1,2):smarthomeroom(1,2):room(2);
 1,0,smarthomeroom(1,3):smarthomeroom(1,3):room(3);
 1,0,smarthomeroom(1,4):smarthomeroom(1,4):room(4)
 }<=2:-smarthome(1).

1<=#count{1,0,smarthomeroom(1,2):
 smarthomeroom(1,2):smarthome(1)}<=1:-room(2).
1<=#count{1,0,smarthomeroom(1,3):
 smarthomeroom(1,3):smarthome(1)}<=1:-room(3).
1<=#count{1,0,smarthomeroom(1,4):
 smarthomeroom(1,4):smarthome(1)}<=1:-room(4).

% further constraints
:- smarthome(1); country(1,austria); not 2<=#count
 {1,0,smarthomeroom(1,2):smarthomeroom(1,2):room(2);
 1,0,smarthomeroom(1,3):smarthomeroom(1,3):room(3);
 1,0,smarthomeroom(1,4):smarthomeroom(1,4):room(4)}<=2.

% customer requirements
room(2). type(2,kitchen).
```

Figure 10.15.: Grounded version of restricted *smarthome* configuration model.

As the customer prefers *room 2* and *country austria* requires 2 rooms, there is only one answer set (*Answer: 1*) with one room - its type is *kitchen* (as specified by the customer requirements) and the *country*

is *germany*. Sketch of the reasoning steps: *room(2)* can be seen as a propositional variable with truth value *TRUE*. It appears in the body of the second rule for generation of associations and that body contains no other variables. Therefore the head is evaluated and requires exactly one variable in the count aggregate to be set to *TRUE*. The only one is *smarthomeroom(1,2)* and it is founded if *smarthome(1)* is generated by the first count aggregate for generation of instances. Therefore, it derives the facts *smarthomeroom(1,2)* and *smarthome(1)*. For *country*, there are exactly two alternatives, but only *germany* does not lead to a constraint violation. Therefore *country(1,germany)* is added as a fact (i.e. assigned truth value *TRUE* in the SAT view of the reasoning process). As no other variables need to be set (all count aggregates are fulfilled), we have the first solution.

All other solutions derive a second room. This allows all combinations of the alternatives for *type (kitchen, livingroom)*, *country (germany, austria)* and *identifier (3, 4)*. One can imagine that the corresponding multiplication of alternatives can lead to tremendously many solutions. At least concerning the identifiers, the solutions are equivalent (i.e. there is no relevant difference between answer sets 2 to 5 and answer sets 6 to 9). In order to reduce the search space for such unneeded solutions, symmetry breaking techniques (Drescher et al., 2010) and search heuristics (Gebser et al., 2013) can be used.

Figure 10.17 shows symmetry breaking constraints which force the solver to use identifiers from lowest to highest. In the knowledge base defined by the ASP entries of Figure 10.2–10.13, the number of answer sets would be reduced from 8.400 to 2.800 if the symmetry breaking constraints are taken into account. Adding constraint *:- room(X), proom(Y), X>Y, not room(Y).* to Figure 10.14 would reduce the number of answer sets from 9 to 5 in Figure 10.16.

```
- Answer: 1 -
smarthome(1) country(1,germany)
room(2) type(2,kitchen)
smarthomeroom(1,2)
- Answer: 2 -
smarthome(1) country(1,germany)
room(2) type(2,kitchen) room(4) type(4,livingroom)
smarthomeroom(1,2) smarthomeroom(1,4)
- Answer: 3 -
smarthome(1) country(1,austria)
room(2) type(2,kitchen) room(4) type(4,livingroom)
smarthomeroom(1,2) smarthomeroom(1,4)
- Answer: 4 -
smarthome(1) country(1,germany)
room(2) type(2,kitchen) room(4) type(4,kitchen)
smarthomeroom(1,2) smarthomeroom(1,4)
- Answer: 5 -
smarthome(1) country(1,austria)
room(2) type(2,kitchen) room(4) type(4,kitchen)
smarthomeroom(1,2) smarthomeroom(1,4)
- Answer: 6 -
smarthome(1) country(1,germany)
room(2) type(2,kitchen) room(3) type(3,kitchen)
smarthomeroom(1,2) smarthomeroom(1,3)
- Answer: 7 -
smarthome(1) country(1,austria)
room(2) type(2,kitchen) room(3) type(3,kitchen)
smarthomeroom(1,2) smarthomeroom(1,3)
- Answer: 8 -
smarthome(1) country(1,germany)
room(2) type(2,kitchen) room(3) type(3,livingroom)
smarthomeroom(1,2) smarthomeroom(1,3)
- Answer: 9 -
smarthome(1) country(1,austria)
room(2) type(2,kitchen) room(3) type(3,livingroom)
smarthomeroom(1,2) smarthomeroom(1,3)
```

Figure 10.16.: Solutions for *smarthome* knowledge base of Figure 10.15.

```
:- room(X), proom(Y), X>Y, not room(Y).
:- stove(X), pstove(Y), X>Y, not stove(Y).
:- tv(X), ptv(Y), X>Y, not tv(Y).
```

Figure 10.17.: Symmetry breaking constraints for the entries of Figures 10.2–10.13.

10.6. AGILE Configuration Technologies

The configuration knowledge representations discussed in this chapter are applied within the scope of the European Union project AGILE[¶] that focuses on the development of recommendation and configuration technologies for IoT gateways. Within AGILE, configuration technologies are applied to support different

[¶] agile-iot.eu.

kinds of ramp-up scenarios, i.e., initial setups of IoT gateways infrastructures entailing the needed hardware and software components. Furthermore, AGILE supports runtime configuration and reconfiguration, for example, in terms of optimizing the usage of data exchange protocols with regard to optimality criteria such as economy and efficiency. The basis for AGILE configuration solutions in ramp-up domains is the *clingo* environment. In AGILE, we are especially focusing on improving the performance of constraint-based reasoning and model-based diagnosis that are both supporting technologies also in the context of answer set programs (Felfernig et al., 2012; Shchekotykhin, 2014; Teppan and Friedrich, 2016).

10.7. Research Issues

There are still a couple of research challenges to be tackled to make ASP-based configuration more applicable and performant. Graphical configuration knowledge representations and a corresponding automated translation into ASP-based representations will help to improve knowledge engineering processes. An automated generation of ASP knowledge bases from object-oriented product topologies has already been proposed in (Falkner et al., 2015); translation routines for standard constraints such as *requires*, *excludes*, and *resources* would help to further increase knowledge engineering efficiency. Improving constraint answer set programming or finding new ways of integrating ASP and CSP could lead to a full exploitation of the different advantages of both paradigms. In order to solve large-sized problems, lazy grounding and heuristics must be combined and improved (related work is reported, e.g., in (Weinzierl, 2017)). All means to reduce problem sizes of ASPs should be exploited, for example, precise estimation of the number of needed instances (see (Taupe et al., 2016)). Finally, domain-specific constraints such as discussed in (Hotz and Wolter, 2013) have to be analyzed with regard to their representation in ASP.

10.8. Conclusions

In this chapter we give an overview of basic applications of configuration technologies in Internet of Things (ioT) scenarios. In this context we show how to apply answer set programming (ASP) techniques to represent and solve configuration problems. ASP is a logic-based approach and well-suited for a component-oriented representation of configuration tasks. This capability is extremely useful especially in large and complex product domains. In order to provide a basic reference for ASP beginners, we show how to represent most representative constraints in ASP.

Chapter 11

Conclusions and Future Work

Many scenarios in which groups of users must take decisions occur in the world today, such as *choosing a restaurant to have a dinner at with family members* or *selecting a movie to watch with friends* (Masthoff, 2011; Felfernig et al., 2018d). This increased interest in group decision scenarios motivated us to improve and further develop group decision technologies. This thesis provides different approaches to boost the quality of group decisions, to counteract biases which deteriorate group decisions and to increase the satisfaction of the group members. These improvements are briefly summarized in this chapter. First, we present our contributions based on the research questions (see Section 11.1), then show the limitations of our approaches, and finally give an outlook on future research issues.

11.1. Conclusions

In this section, we provide a summary of the research questions presented in Section 1.2 and summarize our main contributions.

Research question Q1:

How can constraint-based recommendation for individual users be implemented for group scenarios?
In the context of this work, we provided an overview of group recommendation algorithms and techniques to researchers and practitioners in the field of group recommender systems (see Chapter 2). In particular, we introduced algorithms and techniques based on the recommendation paradigms for individual users. Furthermore, we illustrated group recommendation scenarios through real-world examples and showed how to apply such group scenarios for the well-known recommendation concepts such as *collaborative filtering, content-based filtering, constraint-based including utility-based recommendation, critiquing-based,* and *hybrid recommendation*. These recommendation concepts were classified in two aggregation strategies: *aggregated predictions* and *aggregated models*. In the *aggregated predictions* strategy, there are two basic approaches. (1) Recommendations for individual group members are generated and then merged together for the recommendation of items to groups. (2) Group-member-specific predictions for candidate items are aggregated. The outcome of this approach is a *ranking of candidate items*. In the

aggregated models strategy, the profiles of individual group members are aggregated to generate a group profile. Subsequently, group recommendations are determined based on the aggregated profile.

Research question Q2:

Which recommendation strategy should be applied in which item domain?
To analyze the impact of the item domain on the appropriateness of group recommendations strategies, we conducted a user study where participants explicitly specified their preferences with regard to a set of items (see Chapter 3). For this purpose, we classified item domains according to their decision efforts (i.e., the degree of user involvement) and selected *restaurants* as an example of low-involvement item domains and *shared apartments* as an example of high-involvement item domains. Subsequently, the dataset collected from the user study was analyzed and aggregation heuristics were then used to infer the corresponding group preferences. The following group aggregation functions were analyzed with regard to their prediction qualities: *Most Pleasure, Least Misery, Average Voting, Minimal Group Distance, Ensemble Voting,* and *Multiplicative*. The results showed that user study participants applied different aggregation strategies depending on the underlying item domain.

Research question Q3:

How does recommendation diversity influence the information exchange among group members?
In Chapter 4, we introduced an approach to increase the knowledge exchange among group members for the purpose of discovering the relevant knowledge (i.e., hidden profiles) in the context of a group decision and achieving high-quality decisions through the discovered knowledge exchange among group members. Our focus was to analyze the impact of recommendation diversity on the frequency of information exchange between group members. For this purpose, we analyzed the following recommendation strategies with varying degrees of diversity: *minimum group distance* (i.e., low-diversity recommendations), *maximum group distance* (i.e., high-diversity recommendations), and *average group distance* (i.e., recommendation diversity that represents a compromise between minimum and maximum). Furthermore, we conducted a user study using the group decision support environment CHOICLA (Stettinger and Felfernig, 2014), in order to analyze the impact of these recommendation strategies on the knowledge exchange among group members. Our empirical results showed that preference aggregation mechanisms with a high resulting recommendation diversity increase the frequency of knowledge exchange within a group.

Research question Q4:

How do "Group Polarization Effects" influence the outcome of group decisions and how can these effects be counteracted?
In this context, we analyzed the existence of "Group Polarization Effects" in *risk-* and *cost-related* group decisions. We conducted a user study with 211 computer science students who were involved in individual and group decisions in two different dimensions (*risk analysis* and *cost estimation*). Subsequently, we analyzed the collected dataset to investigate whether there was a tendency of the groups to make decisions that are more extreme than the average of individual group members preferences. The results showed the following outcomes (see Chapter 5): In the *risk* dimension, "Group Polarization Effects" only occur if individual group members tend to make cautious decisions, whereas in the *cost estimation* dimension, these effects only exist at the lower boundaries of the cost range. In addition, we presented a solution to

counteract this bias by diversifying the preferences of group members. This approach helps to avoid the negative impacts of this bias on the decision quality.

Research question Q5:

Which aggregation functions are suitable for predicting items to groups in situations where the preferences of group members become inconsistent?
Chapter 6 introduced a constraint-based group recommender system, which focused on comparing aggregation functions with regard to their capability to predict relevant items to groups. In particular, we analyzed the prediction quality of aggregation functions in situations where no solution could be found for a given set of preferences. In this context, our proposed recommender system attempts to suggest items with a high related decision effort to groups (i.e., high-involvement items). Two categories of aggregation functions were analyzed for determining the prediction quality: *consensus-based* (Average Voting, Minimal Group Distance, Multiplicative, and Ensemble Voting) and *borderline* (Least Misery and Most Pleasure). These aggregation functions were analyzed based on a dataset collected in a user study with 263 participants. The outcome shows that consensus-based aggregation functions which consider all group members preferences achieved a higher prediction quality compared to borderline aggregation functions which solely focus on the preferences of some individual group members.

Research question Q6:

How to identify a socially-aware diagnosis when group members' preferences are inconsistent?
In the context of constraint-based group recommenders, we presented a guided approach that determines socially-aware diagnoses based on different aggregation functions in situations where the preferences of group members are inconsistent with the underlying constraint set (see Chapter 7). The goal of this approach was to identify diagnoses for groups that best match the preferences of all group members (i.e., we tried to identify a diagnosis which was fair and did not decrease the overall group satisfaction). For this purpose, we conducted a user study where participants first articulated their preferences for a digital camera (i.e., high-involvement item), and then selected a digital camera.* We then developed a constraint-based recommender system which suggests socially-aware diagnoses guided by the following aggregation functions: *Least Misery, Most Pleasure, Average Voting,* and *Majority*. Finally, based on the collected dataset (i.e., the preferences of groups and their diagnoses), aggregation functions were evaluated with regard to their prediction qualities. Our results showed that the aspect of *fairness* (i.e., the idea of *Least Misery* aggregation function) plays a major role in the selection of high-involvement items. Furthermore, we could show that the prediction quality of our approach outperforms the basic approaches, such as *Breadth-First Search* and *Direct Diagnosis*.

Research question Q7:

How can a constraint-based recommender system identify a recommendation which is similar to the preferences articulated by a user?
Chapter 8 introduced two constraint-based recommendation approaches which provide similar recommendations to users based on their articulated preferences. As already stated in previous sections,

*The preferences of each user study participant were inconsistent with the underlying knowledge base.

such recommender systems are usually applied in complex domains where millions of recommendation possibilities exist. The goal of our approach was to identify a recommendation from a long list of recommendations similar to preferences articulated by the users. We tested our approach with datasets from two domains (*bike* and *personal computer*) and performed an evaluation of runtime and the similarity degree between users' requirements and the identified recommendation. The results of our evaluation indicate that our approaches are able to recommend similar items in an effective and efficient way.

Research question Q8:

How can requirement dependencies be identified automatically using supervised classification techniques?
Since the size and complexity of software projects is rapidly increasing, the identification of the dependencies between requirements has become a challenging task for humans. Due to this fact, we realized that there was an urgent need for automated techniques which can assist stakeholders in finding dependencies between requirements. We developed an approach which can detect requirement dependencies of the type *"requires"* since it is the dependency type that most frequently occurs in software projects (Ferber et al., 2002). For this purpose, we conducted a user study that focused on the detection of dependencies between text-based requirements for a *sports watch* (see Chapter 9). Subsequently, the collected dataset was used to train and test our system by using several classification approaches (*Naive Bayes, Linear Support Vector Classifier, k-Nearest Neighbors Classifier*, and *Random Forest*). Our approach analyzed the *title* and the *description* of each requirement and identified dependencies by using *Natural Language Processing* techniques. Furthermore, for this approach, we first prepared the collected dataset (*preprocessing, tokenization, Part-Of-Speech tagging*), and then analyzed our approach with different classifiers. The results showed that the *Random Forest classifier* could correctly detect new dependencies between requirements with an F_1 score of 0.82.

Research question Q9:

How can configuration knowledge be efficiently represented in the IoT domain?
In Chapter 10, we introduced the application of an efficient knowledge representation for complex configuration scenarios in the *Internet of Things* (IoT) domain. Since the size and complexity of configuration problems in the IoT domain increases dramatically, the basic knowledge representation and reasoning such as *constraint satisfaction* are not easy to use and maintain (Hotz et al., 2014; Felfernig et al., 2014a). This challenge motivated us for the application of an efficient knowledge representation. We applied *Answer Set Programming* (ASP) in the IoT domain, which is a logic-based approach and well-suited for component-oriented knowledge representations (Gelfond and Lifschitz, 1988; Soininen and Niemelä, 1998). In this context, we showed the applicability of ASP representations in the IoT domain and provided basic configuration examples. Furthermore, we demonstrated the limitations of ASP and discussed open issues for future research that will help to accelerate a broad application of ASP technologies in knowledge-based configuration.

Limitations

This thesis presented several approaches to support high-quality group decisions. However, our approaches still have some limitations in the evaluation methods and decision making processes. Our approaches were evaluated using small datasets for example (datasets from 200-400 participants) collected from homogenous user study participants (computer science students with $\sim$ 90% male and $\sim$ 10% female proportion). In addition, our user studies were conducted with individual participants. Due to the fact that our approaches are intended for application in group scenarios, we synthesized these datasets to generate a dataset for groups (i.e., groups were synthetically generated). In this context, we clustered participants with similar preferences (i.e., homogeneous groups were generated) based on the reason that group decisions are usually made in groups consisting of group members with similar tastes (e.g., friends, family members, etc.). Many decision scenarios arise, however, in which items are consumed by diverse group members (e.g., *restaurant recommendations for the participants of a conference* or *music recommendations for users who are currently in the fitness studio*). Another limitation lies in the consideration of the factors which can influence group decisions, such as *fairness in repeated group decisions*, *the impact of group diversity on decision making processes*, *group dynamics*, *cohesiveness between group members*, and many other factors. Moreover, we did not analyze the impact of *intelligent user interface elements* on group decisions, which could accelerate the *consensus making process*, promote *group discussions*, or disclose *hidden profiles*. Further limitations are the investigation of *negotiation patterns* to accelerate the consensus process within the group (i.e., to resolve conflicts) and *persuasive explanations* to increase the acceptance of the recommended item.

11.2. Future Work

This section presents relevant topics for future research based on the limitations defined in Section 11.1.

Intelligent user interfaces

In this thesis, we focused primarily on the determination of optimal strategies (i.e., heuristics) for making group recommendations in dependence on both the decision scenario and the item domain. There are other factors, however, which can influence the quality of group decisions. For instance, the selection of *preference acquisition interfaces* (5-star rating vs. ranking vs. thumbs up/down, drag&drop vs. emoticons vs. textual explanations) can have an impact on how group members articulate their preferences for items (McNee et al., 2003; Chen, 2011). These interfaces should thus be selected based on the item domain, group dynamics, and the decision scenario. Should this not be done, there is a risk that the articulated preferences of the group members will not truly reflect their opinions. Moreover, such interfaces can be applied for increasing the frequency of information exchanges between group members. For instance, showing *notification icons* which refer to the occurred conflicts among group members can help group members be aware of the existence of the conflicts in early stages of decision making processes and thus, additional group discussions can be triggered (i.e., disclosing of hidden profiles). Another possibility is the integration of *gamification-based* concepts in recommender systems such as *Planning Poker* (Haugen, 2006), which increases the participation of users in group decisions and accelerates the consensus making process (Felfernig et al., 2017c). To the best of our knowledge, related work does not exist concerning the inclusion of the intelligent user interfaces in group decision making processes. For future work, we propose innovative decision making approaches, which support the discussed user interfaces and analyze the impacts of these user interfaces in the context of different item domains and decision scenarios.

Group diversity and group dynamics

Group diversity: As already stated in Section 11.1, we clustered participants with similar preferences and analyzed our approaches with homogeneous groups (e.g., similar ages and educational backgrounds). Within the scope of future work, we will further evaluate our recommendation approaches with *diverse groups* (i.e., group members with diverse cultures, educational backgrounds, genders, and ages). In (Atas et al., 2017), we found that group diversity can help to increase the knowledge exchange among group members and thus also raises the quality of the group decision. The answer to the question on the optimal degree of *information exchange* and *perceived recommendation quality*, however, has until now remained unclear. In this context, the optimal degree of group diversity should be determined depending on the decision scenario. Diverse groups usually consist of group members with different tastes, interests, occupations, cultures, and educational backgrounds. In such groups, members analyze a group decision from different perspectives and this triggers additional discussions which help to disclose the hidden profiles. Therefore, (new) group members with diverse preferences need to be integrated into homogeneous groups in order to provide other perspectives and to increase the decision quality (Zhang et al., 2007). In this context, the question arises of *"how to identify correct persons/group members (i.e., experts) who are able to make an optimal group decision"*. In order to support such scenarios, *Liquid Democracy* concepts can be applied which represent a hybrid voting model of participative democracy (Zhang and Zhou, 2017; Atas et al., 2018d). Besides, to achieve an optimal group recommendation, each group member should be an expert in the corresponding item domain. Unfortunately, in reality, group members do not always have the necessary knowledge about the corresponding item domain and therefore, the advice of the experts is necessary to precisely evaluate items. In such situations, *Liquid Democracy* concepts enable group members to play an active role (i.e., the current user evaluates an item by him/her-self) or a passive role (i.e., the current user asks for advice of an expert). In this context, our wish is to develop methods and approaches for determining the optimal degree of group diversity and to analyze the impact of different group diversity degrees on decision making. Furthermore, we intend to integrate *Liquid Democracy* in our group recommender systems.

Group dynamics: In reality, group decisions are influenced by several factors which are often not considered in group recommender systems, such as the *interaction* among group members (i.e., *group dynamics*) or the *cohesiveness* between group members which is helpful for understanding the decision making behaviour of groups. Group dynamics (Forsyth, 2006; Brown, 2012) describe the social interaction of groups and this phenomenon is classified into *intragroup dynamics* and *intergroup dynamics*. The former refers to the interaction between group members within a group, whereas the latter is the interaction of a group with other groups. In the context of group decision making, the *intragroup dynamics* affect group decisions in situations *in which conflicts occur in the context of group member preferences, which they then attempt to resolve in the process of negotiating to reach a consensus,* or *when they exchange relevant information about a decision to disclose hidden profiles*. In this thesis, we have mostly considered the preferences of individual group members, but have not considered the impact of such factors on different stages of the group decision making process. Bruce W. Tuckman introduced in his study the stages in the building of groups (e.g., *forming, storming, norming,* and *performing*) and showed that these stages are necessary for groups to grow, face challenges, overcome problems, find solutions, and deliver results (Tuckman, 1965; Tuckman and Jensen, 1977). In the context of group decision technologies, the group dynamics can influence group-building stages in the following scenarios: *forming* (when groups are formed for the first time, group members begin to know each other's preferences), *storming* (diverse

preferences of group members have the effect of raising conflicts among group members), *norming* (groups begin to resolve conflicts and make consensus), and *performing* (a group decision is made). Group dynamics clearly influence the decisions made in each of the group forming stages. For future work, we thus plan to investigate the group dynamics in our approaches and also to analyze the influence these have on different domains and scenarios.

Social factors in group decisions
In the previous paragraphs, we pointed out that the application of intelligent user interfaces, diversity of the group, and group dynamics can have a strong impact on decision making behaviour. In this context, we further discuss the influence of some factors which were partly considered in this thesis. For instance, in group scenarios and repeated decisions, *fairness aspects* can affect the satisfaction of group members (Burke, 2017; Xiao et al., 2017; Felfernig et al., 2018b). In the context of group recommendations, fairness should be taken into account in the following scenarios: *conflict resolution* (the identified diagnosis must be fair to all group members and should not require only the adaption or removal of the constraints of some group members), *group recommendation generation* (the applied aggregation function should consider the preferences of all group members and identify a recommendation which satisfies all group members), and *repeated decisions* (if a decision is often repeated by the same group, then the preferences of group members should be taken into account on an equal basis, i.e., no member has a higher priority than the others).

Moreover, in our thesis, we did not consider the *personality* and *emotions* of group members on the group decision making process (Tkali et al., 2016). It is an acknowledged fact that the personality and current emotions of a user can influence his/her decision making behaviour. For instance, Neidhardt et al. introduced an approach to elicit the personality information of users in the tourism domain (Neidhardt et al., 2015). Recently, researchers have tried to model the personality and emotion of users by integrating this information into their systems as a means of understanding the human personality and emotions (Kilmann and Thomas, 1977; D'Errico and Poggi, 2016; Felfernig et al., 2018c). In the future, we should consider all of the factors mentioned in order to increase the decision quality and to better understand the decision making behaviour of users.

Conflict detection and negotiation mechanisms
Although decision technologies have been applied for a number of years, there are still open issues in the context of basic decision making processes. For instance, the basic question of *when is a conflict seen as a conflict?* has only been partly answered to date. We assume that conflict detection depends on many factors, such as the *personalities* of involved group members (e.g., optimistic or pessimistic characteristics), the *item domain*, and the *applied rating scale* (e.g., in 5-star rating scale, a difference higher than one star can be seen as a conflict, where in percentage scale, a difference of more than 15% can be seen as a conflict). Unfortunately, the determination of conflicts in group decisions scenarios is very challenging. To the best of our knowledge, no related work exists which can answer this question.

Another important future research topic is the investigation of the *negotiation mechanisms*. Traditionally, the members of groups apply different negotiation types (also termed *negotiation patterns*) to provide arguments for their conflict resolution proposals (Brett, 1991; Kersten, 1997; Boehm et al., 2001; Salamo et al., 2012). The application of the negotiation patterns in repeated group decision making in particular can accelerate the consensus making process and increase the acceptance and satisfaction of all group

members with regard to the recommended items. These patterns usually consist of promises for the future, which can be differentiated based on the duration of the negotiation process (*long-term* vs. *short-term*). For instance, in the context of a group decision on *restaurant selection* which is repeated every week, the following negotiation pattern can be used: *"I agree to visit restaurant X if we agree to visit restaurant Y next time!"*. Furthermore, we can differentiate these negotiation types based on the following categories: *threat* (e.g., *"If we select the restaurant X, then I will not come, because I don't like Italian food."*), *reward* (e.g., *''If we select my favorite restaurant now, then next time, we can have dinner in your favorite restaurant."*), *appeal* (e.g., *"We should select restaurant X because nobody has a problem with this."*). This means, we can classify the negotiation types into six patterns (*short-term award, long-term award, short-term threat, long-term threat, short-term appeal,* and *long-term appeal*). In this context, explanations for group recommendations can be shaped using these negotiation patterns, which provide an insight into the group recommendation process and help to speed up the consensus making process. However, up to now, these negotiation patterns have not been adequately analyzed and integrated into decision making support systems. Within the scope of future work, we will analyze the impact of negotiation patterns on group decision processes.

List of Figures

List of Tables

Bibliography

T. Abeel, Y. V. de Peer, and Y. Saeys. 2009. Java-ML: A Machine Learning Library. *The Journal of Machine Learning Research* 10, 931–934 (April 2009). (Cited on page 70.)

G. Adomavicius, J. Bockstedt, S. Curley, and J. Zhang. 2011. Recommender Systems, Consumer Preferences, and Anchoring Effects. In *RecSys 2011 Workshop on Human Decision Making in Recommender Systems*. 35–42. (Cited on pages 4, 62, and 66.)

M. Aldanondo and E. Vareilles. 2008. Configuration for Mass Customization: How to Extend Product Configuration Towards Requirements and Process Configuration. *Journal of Intelligent Manufacturing* 19, 5 (2008), 521–535. (Cited on page 5.)

L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso. 2003. Intrigue: Personalized Recommendation of Tourist Attractions for Desktop and Hand Held Devices. *Applied Artificial Intelligence* 17, 8-9 (2003), 687–714. (Cited on page 74.)

K. Arrow. 1950. The Difficulty in the Concept of Social Welfare. *Journal of Political Economy* 58, 4 (1950), 328–346. (Cited on page 22.)

M. Atas, A. Felfernig, M. Stettinger, and T. N. T. Tran. 2017. Beyond Item Recommendation: Using Recommendations to Stimulate Knowledge Sharing in Group Decisions. In *9th International Conference on Social Informatics (SocInfo 2017)*. Oxford, UK, 368–377. (Cited on pages 3, 12, 57, and 142.)

M. Atas, S. Reiterer, A. Felfernig, T. N. T. Tran, and M. Stettinger. 2018a. Polarization Effects in Group Decisions. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP '18)*. ACM, New York, NY, USA, 305–310. (Cited on pages 4, 13, and 65.)

M. Atas, R. Samer, and A. Felfernig. 2018b. Automated Identification of Type-Specific Dependencies between Requirements. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 688–695. (Cited on pages 5, 15, and 109.)

M. Atas, R. Samer, A. Felfernig, T. N. T. Tran, S. Polat-Erdeniz, and M. Stettinger. 2019a. Socially-Aware Diagnosis for Constraint-Based Recommendation. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '19)*. ACM, New York, NY, USA, 121–129. (Cited on pages 3, 14, and 83.)

M. Atas, T. N. T. Tran, A. Felfernig, S. Polat-Erdeniz, R. Samer, and M. Stettinger. 2019b. Towards Similarity-Aware Constraint-Based Recommendation. In *Advances and Trends in Artificial Intelligence. From Theory to Practice*, F. Wotawa, G. Friedrich, I. Pill, R. Koitz-Hristov, and M. Ali (Eds.). Springer International Publishing, Cham, 287–299. (Cited on pages 3, 15, and 97.)

M. Atas, T. N. T. Tran, A. Felfernig, and R. Samer. 2018c. Socially-Aware Recommendation for Over-Constrained Problems. In *Recent Trends and Future Technology in Applied Intelligence*, M. Mouhoub,

S. Sadaoui, O. Ait Mohamed, and M. Ali (Eds.). Springer International Publishing, Cham, 267–278. (Cited on pages 3, 13, and 73.)

M. Atas, T. N. T. Tran, R. Samer, A. Felfernig, M. Stettinger, and D. Fucci. 2018d. Liquid Democracy in Group-based Configuration. In *Proceedings of the 20th Configuration Workshop, Graz, Austria, September 27-28, 2018*. 93–98. (Cited on page 142.)

L. Atzori, A. Iera, and G. Morabito. 2010. The Internet of Things: A Survey. *Computer Networks* 54, 15 (2010), 2787–2805. (Cited on pages 6 and 123.)

A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. 2013. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM conference on Recommender systems*. 121–128. (Cited on page 1.)

L. Baltrunas, T. Makcinskas, and F. Ricci. 2010. Group Recommendations with Rank Aggregation and Collaborative Filtering. In *4th ACM Conference on Recommender Systems*. Barcelona, Spain, 119–126. (Cited on pages 20, 21, 74, 75, and 90.)

A. Bangor, P. T. Kortum, and J. T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of HumanComputer Interaction* 24, 6 (2008), 574–594. (Cited on page 87.)

S. Berkovsky and J. Freyne. 2010. Group-based Recipe Recommendations: Analysis of Data Aggregation Strategies. In *4th ACM Conference on Recommender Systems*. Barcelona, Spain, 111–118. (Cited on pages 4, 21, and 43.)

S. Berkovsky, J. Freyne, M. Coombe, and D. Bhandari. 2010. Recommender Algorithms in Activity Motivating Games. *ACM Conference on Recommender Systems (RecSys'10)* (2010), 175–182. (Cited on pages 25 and 28.)

B. Boehm, P. Grunbacher, and R. O. Briggs. 2001. Developing groupware for requirements negotiation: lessons learned. *IEEE Software* 18, 3 (May 2001), 46–55. (Cited on page 143.)

M. U. Bokhari and S. T. Siddiqui. 2010. A Comparative Study of Software Requirements Tools For Secure Software Development. 2, 2 (2010), 1–12. (Cited on page 109.)

L. Boratto and S. Carta. 2015. The Rating Prediction Task in a Group Recommender System that Automatically Detects Groups: Architectures, Algorithms, and Performance Evaluation. *Journal of Intelligent Information Systems* 45, 2 (2015), 221–245. (Cited on pages 21, 58, and 152.)

L. Boratto, S. Carta, and G. Fenu. 2017. Investigating the Role of the Rating Prediction Task in Granularity-based Group Recommender Systems and Big Data Scenarios. *Information Sciences* 378 (2017), 424–443. (Cited on pages 19 and 21.)

J. Borràs, A. Moreno, and A. Valls. 2014. Intelligent tourism recommender systems: A survey. *Expert Systems with Applications* 41, 16 (2014), 7370–7389. (Cited on page 1.)

K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer. 2011. SMOTE: Synthetic Minority Over-sampling Technique. *CoRR* abs/1106.1813 (2011). (Cited on pages 114 and 121.)

L. Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (01 Oct 2001), 5–32. (Cited on page 116.)

J. M. Brett. 1991. Negotiating group decisions. *Negotiation Journal* 7, 3 (01 Jul 1991), 291–310. (Cited on page 143.)

D. Bridge, M. H. Göker, L. McGinty, and B. Smyth. 2005. Case-based recommender systems. *The Knowledge Engineering Review* 20, 3 (2005), 315–320. (Cited on pages 2 and 8.)

M. Brocco and G. Groh. 2009. Team Recommendation in Open Innovation Networks. In *ACM Conference on Recommender Systems (RecSys'09)*. NY, USA, 365–368. (Cited on page 47.)

R. Brown. 2012. *Group Processes*. Blackwell Publishing. (Cited on page 142.)

R. Burke. 2000. Knowledge-based recommender systems. *Encyclopedia of library and information systems* 69, Supplement 32 (2000), 175–186. (Cited on pages 1, 2, 83, 97, and 98.)

R. Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction (UMUAI)* 12, 4 (2002), 331–370. (Cited on pages 2, 3, and 43.)

R. Burke. 2017. Multisided Fairness for Recommendation. In *2017 Workshop on Fairness, Accountability, and Transparency in Machine Learning (FAT/ML 2017)*. 1–5. (Cited on page 143.)

R. Burke, A. Felfernig, and M. Goeker. 2011. Recommender Systems: An Overview. *AI Magazine* 32, 3 (2011), 13–18. (Cited on page 1.)

M. Butkiewicz, R. Mueller, D. Selic, E. Dawson, and J. Meiler. 2009. Application of Machine Learning Approaches on Quantitative Structure Activity Relationships. (Cited on page 118.)

C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher. 2009. A Recommender System for Requirements Elicitation in Large-scale Software Projects. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC '09)*. ACM, New York, NY, USA, 1419–1426. (Cited on page 110.)

L. Chen, G. Chen, and F. Wang. 2015. Recommender Systems Based on User Reviews: the State of the Art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154. (Cited on pages 20 and 21.)

L. Chen, M. deGemmis, A. Felfernig, P. Lops, F. Ricci, and G. Semeraro. 2013. Human Decision Making and Recommender Systems. *ACM Transactions on Interactive Intelligent Systems* 3, 3 (2013), 1–7. (Cited on page 58.)

L. Chen and P. Pu. 2012a. Critiquing-based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction (UMUAI)* 22, 1–2 (2012), 125–150. (Cited on page 39.)

Y. Chen. 2011. Interface and Interaction Design for Group and Social Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys 11)*. Association for Computing Machinery, New York, NY, USA, 363366. (Cited on page 141.)

Y. Chen and P. Pu. 2012b. CoFeel: Using emotions for social interaction in group recommender systems. *CEUR Workshop Proceedings* 891 (01 2012), 48–55. (Cited on page 58.)

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. 2007. A Short Introduction to Computational Social Choice. In *33rd conference on Current Trends in Theory and Practice of Computer Science*. Harrachov, Czech Republic, 51–69. (Cited on pages 22, 24, and 149.)

K. Christakopoulou, F. Radlinski, and K. Hofmann. 2016. Towards Conversational Recommender Systems. In *International Conference on Knowledge Discovery and Data Mining (KDD 2016)*. San Francisco, CA, USA, 815–824. (Cited on pages 20 and 21.)

C. Christakou, S. Vrettos, and A. Stafylopatis. 2007. A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools* 16, 05 (2007), 771–792. (Cited on page 1.)

M. Dabrowski and T. Acton. 2011. Beyond Similarity-Based Recommenders: Preference Relaxation and Product Awareness. In *E-Commerce and Web Technologies*, C. Huemer and T. Setzer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 296–307. (Cited on page 102.)

Z. Dávid. 2016. Recommender Systems meet Finance: a Literature Review. In *FINREC*. 3–10. (Cited on page 1.)

J. de Kleer, A. K. Mackworth, and R. Reiter. 1992. Readings in Model-based Diagnosis. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter Characterizing Diagnoses and Systems, 54–65. (Cited on page 98.)

E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. 1988. Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics* 44, 3 (1988), 837–845. (Cited on page 117.)

T. DePessemier, J. Dhondt, and L. Martens. 2017. Hybrid Group Recommendations for a Travel Service. *Multimedia Tools and Applications* 76, 2 (2017), 2787–2811. (Cited on page 43.)

T. DePessemier, J. Dhondt, K. Vanhecke, and L. Martens. 2015. TravelWithFriends: A Hybrid Group Recommender System for Travel Destinations. In *9th ACM Conference on Recommender Systems, Workshop on Tourism Recommender Systems*. 51–60. (Cited on page 43.)

T. DePessemier, S. Dooms, and L. Martens. 2013. An Improved Data Aggregation Strategy for Group Recommenders. In *3rd Workshop on Human Decision Making and Recommender Systems (held in conjunction with the 7th ACM Conference on Recommender Systems)*. Hong Kong, China, 36–39. (Cited on page 21.)

T. DePessemier, S. Dooms, and L. Martens. 2014. Comparison of Group Recommendation Algorithms. *Multimedia Tools and Applications* 72, 3 (2014), 2497–2541. (Cited on page 43.)

F. D'Errico and I. Poggi. 2016. Social Emotions. A Challenge for Sentiment Analysis and User Models. Springer, 13–34. (Cited on page 143.)

C. Drescher, O. Tifrea, and T. Walsh. 2010. Symmetry-breaking answer set solving. In *ICLP10 Workshop on Answer Set Programming and Other Computing Paradigm*. 177–194. (Cited on page 134.)

H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli. 2011. On-demand Feature Recommendations Derived from Mining Public Product Descriptions. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. ACM, New York, NY, USA, 181–190. (Cited on page 5.)

T. Eiter, E. Erdem, H. Erdoğan, and M. Fink. 2009. Finding Similar or Diverse Solutions in Answer Set Programming. In *Proceedings of the 25th International Conference on Logic Programming (ICLP '09)*. Springer-Verlag, Berlin, Heidelberg, 342–356. (Cited on page 98.)

T. Eiter, T. Kaminski, C. Redl, and A. Weinzierl. 2018. Exploiting Partial Assignments for Efficient Evaluation of Answer Set Programs with External Source Access. *Journal of Artificial Intelligence Research* 62 (07 2018), 665–727. (Cited on page 132.)

M. Ekstrand, J. Riedl, and J. Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends in Human-Computer Interaction* 4, 2 (2011), 81–173. (Cited on page 25.)

J. Esser. 1998. Alive and well after 25 Years: A Review of Groupthink Research. *Organizational Behavior and Human Decision Processes* 73, 2-3 (1998), 116–141. (Cited on pages 4 and 67.)

A. Falkner, A. Ryabokon, G. Schenner, and K. Shchekotykhin. 2015. OOASP: Connecting Object-Oriented and Logic Programming. In *13th International Conference on Logic Programming and Nonmonotonic Reasoning*. Lexington, KY, USA, 332–345. (Cited on pages 124 and 136.)

A. Falkner, G. Schenner, G. Friedrich, and A. Ryabokon. 2012. Testing Object-Oriented Configurators With ASP. In *ECAI 2012 Workshop on Configuration*. Montpellier, France, 21–26. (Cited on page 124.)

A. Felfernig. 2014. Biases in Decision Making. In *Proceedings of the International Workshop on Decision Making and Recommender Systems 2014 (CEUR Proceedings)*, Vol. ISSN 1613-0073, Vol 1278. CEUR Proceedings, 32–37. (Cited on pages 7, 58, and 65.)

A. Felfernig, M. Atas, T. N. T. Tran, and M. Stettinger. 2016. *Towards Group-based Configuration*. International Workshop on Configuration 2016, Toulouse, France, 69–72. (Cited on pages 36, 38, and 85.)

A. Felfernig, M. Atas, T. N. T. Tran, M. Stettinger, and S. Polat-Erdeniz. 2017a. 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)*. Arras, France, 335–344. (Cited on pages 2, 4, 12, 22, 49, 59, 65, 72, 73, 83, 88, 95, and 97.)

A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalčič. 2018a. *Algorithms for Group Recommendation*. Springer International Publishing, Cham, 27–58. (Cited on pages 11 and 19.)

A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalčič. 2018b. *Evaluating Group Recommender Systems*. Springer International Publishing, Cham, 59–71. (Cited on page 143.)

A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalčič. 2018c. *Personality, Emotions, and Group Dynamics*. Springer International Publishing, Cham, 157–167. (Cited on page 143.)

A. Felfernig and R. Burke. 2008. Constraint-based Recommender Systems: Technologies and Research Issues. In *ACM International Conference on Electronic Commerce (ICEC08)*. Innsbruck, Austria, 17–26. (Cited on pages 2, 8, 20, 21, and 32.)

A. Felfernig, A. Falkner, M. Atas, S. Polat-Erdeniz, C. Uran, and P. Azzoni. 2017b. ASP-based Knowledge Representations for IoT Configuration Scenarios. In *19th International Configuration Workshop*. 62–67. (Cited on pages 6, 16, and 123.)

A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner. 2004. Consistency-based Diagnosis of Configuration Knowledge Bases. *Artificial Intelligence* 152, 2 (2004), 213–234. (Cited on pages 9, 36, and 84.)

A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. 2011a. Developing constraint-based recommenders. In *Recommender systems handbook*. Springer, 187–215. (Cited on page 2.)

A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. 2014a. *Knowledge-based Configuration: From Research to Business Cases*. Elsevier/Morgan Kaufmann Publishers. 41–72 pages. (Cited on pages 5, 6, 10, 123, 124, and 140.)

A. Felfernig, K. Isak, K. Szabo, and P. Zachar. 2007. The VITA Financial Services Sales Support Environment. Vancouver, Canada, 1692–1699. (Cited on pages 1 and 6.)

A. Felfernig, M. Jeran, G. Ninaus, F. Reinfrank, and S. Reiterer. 2013a. Toward the Next Generation of Recommender Systems. In *Multimedia Services in Intelligent Environments: Recommendation Services*. Springer, 81–98. (Cited on page 19.)

A. Felfernig, M. Mairitsch, M. Mandl, M. Schubert, and E. Teppan. 2009a. Utility-Based Repair of Inconsistent Requirements. In *Next-Generation Applied Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 162–171. (Cited on page 8.)

A. Felfernig, M. Schubert, G. Friedrich, M. Mandl, M. Mairitsch, and E. Teppan. 2009b. Plausible Repairs for Inconsistent Requirements. In *21st International Joint Conference on Artificial Intelligence (IJCAI'09)*. Pasadena, CA, 791–796. (Cited on pages 33, 36, 38, and 39.)

A. Felfernig, M. Schubert, and S. Reiterer. 2013b. Personalized Diagnosis for Over-Constrained Problems. In *23rd International Conference on Artificial Intelligence (IJCAI 2013)*. Peking, China, 1990–1996. (Cited on pages 39, 84, 86, 94, and 101.)

A. Felfernig, M. Schubert, and C. Zehentner. 2012. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)* 26, 1 (2012), 53–62. (Cited on pages 36 and 136.)

A. Felfernig, M. Stettinger, L. Boratto, and M. Tkalcic. 2018d. *Group Recommender Systems: An Introduction*. Springer US. (Cited on pages 2, 3, 4, 65, 83, 84, and 137.)

A. Felfernig, M. Stettinger, A. Falkner, M. Atas, X. Franch, and C. Palomares. 2017c. OPENREQ: Recommender Systems in Requirements Engineering. In *RS-BDA17*. Graz, Austria, 1–4. (Cited on pages 5, 109, and 141.)

A. Felfernig, M. Stettinger, and G. Leitner. 2015. Fostering Knowledge Exchange using Group Recommendations. (Cited on pages 59 and 60.)

A. Felfernig, M. Stettinger, G. Ninaus, M. Jeran, S. Reiterer, A. Falkner, G. Leitner, and J. Tiihonen. 2014b. Towards Open Configuration. In *16th Intl Workshop on Configuration*. Novi Sad, Serbia, 89–94. (Cited on page 5.)

A. Felfernig, C. Zehentner, G. Ninaus, H. Grabner, W. Maalej, D. Pagano, L. Weninger, and F. Reinfrank. 2011b. Group Decision Support for Requirements Negotiation. *Springer Lecture Notes in Computer Science* 7138 (2011), 105–116. (Cited on pages 36 and 58.)

S. Ferber, J. Haag, and J. Savolainen. 2002. Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line. In *Software Product Lines*, G. J. Chastek (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 235–256. (Cited on pages 110 and 140.)

D. Forsyth. 2006. *Group Dynamics*. Thomson Higher Education. (Cited on page 142.)

G. Friedrich, A. Ryabokon, A. Falkner, A. Haselböck, G. Schenner, and H. Schreiner. 2011. (Re)configuration using Answer Set Programming. In *Workshop on Configuration*. Barcelona, Spain, 17–25. (Cited on page 124.)

R. T. Futrell, L. I. Shafer, and D. F. Shafer. 2001. *Quality Software Project Management*. Prentice Hall PTR, Upper Saddle River, NJ, USA. (Cited on pages 68 and 72.)

M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *Trans. Sys. Man Cyber Part C* 42, 4 (July 2012), 463–484. (Cited on pages 114 and 121.)

H. Garcia-Molina, G. Koutrika, and A. Parameswaran. 2011. Information Seeking: Convergence of Search, Recommendations, and Advertising. *Commun. ACM* 54, 11 (2011), 121–130. (Cited on page 21.)

M. Gasparic and A. Janes. 2016. What Recommendation Systems for Software Engineering Recommend. *J. Syst. Softw.* 113, C (March 2016), 101–113. (Cited on page 97.)

M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. 2012. *Answer Set Solving in Practice*. Morgan & Claypool Publishers. (Cited on page 131.)

M. Gebser, B. Kaufmann, R. Otero, J. Romero, T. Schaub, and P. Wanko. 2013. Domain-specific heuristics in answer set programming. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013* (01 2013), 350–356. (Cited on page 134.)

M. Gebser, A. Ryabokon, and G. Schenner. 2015. Combining Heuristics for Configuration Problems Using Answer Set Programming. In *13th International Conference on Logic Programming and Nonmonotonic Reasoning*. Lexington, KY, USA, 384–397. (Cited on page 132.)

M. Gelfond and V. Lifschitz. 1988. The stable model semantics for logic programming. In *5th International Conference of Logic Programming (ICLP'88)*. 1070–1080. (Cited on pages 124 and 140.)

S. Ghazarian and M. Nematbakhsh. 2015. Enhancing Memory-based Collaborative Filtering for Group Recommender Systems. *Expert Systems with Applications* 42, 7 (2015), 3801–3812. (Cited on page 25.)

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–71. (Cited on pages 2 and 83.)

T. Greitemeyer and S. Schulz-Hardt. 2003. Preference-consistent evaluation of information in the hidden profile paradigm: Beyond group-level explanations for the dominance of shared information in group decisions. *Journal of personality and social psychology* 84, 2 (2003), 322. (Cited on pages 7 and 12.)

J. Guo, L. Sun, W. Li, and T. Yu. 2017. Applying Uncertainty Theory to Group Recommender Systems Taking Account of Experts Preferences. *Multimedia Tools and Applications* (2017), 1–18. (Cited on page 38.)

F. Guzzi, F. Ricci, and R. Burke. 2011. Interactive Multi-party Critiquing for Group Recommendation. In *5th ACM Conference on Recommender Systems*. Chicago, IL, USA, 265–268. (Cited on pages 39 and 41.)

M. A. Hall. 1999. Correlation-based feature selection for machine learning. (1999). (Cited on page 114.)

N. C. Haugen. 2006. An empirical study of using planning poker for user story estimation. In *AGILE 2006 (AGILE'06)*. 9 pp.–34. (Cited on page 141.)

E. Hebrard, B. Hnich, B. O'Sullivan, and T. Walsh. 2005. Finding Diverse and Similar Solutions in Constraint Programming. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 1 (AAAI'05)*. AAAI Press, 372–377. (Cited on page 98.)

J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 5–53. (Cited on pages 53, 73, and 83.)

H. F. Hofmann and F. Lehner. 2001. Requirements engineering as a success factor in software projects. *IEEE Software* 18, 4 (Jul 2001), 58–66. (Cited on pages 5, 10, and 109.)

S. Hong, C. Mao, Z. Yang, and H. Lai. 2014. A New Team Recommendation Model with Applications in Social Network. In *18th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. NY, USA, 644–648. (Cited on page 47.)

L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, and K. Wolter. 2014. Configuration Knowledge Representation and Reasoning. In *Knowledge-Based Configuration: From Research to Business Cases*. 41–72. (Cited on pages 10, 124, and 140.)

L. Hotz and K. Wolter. 2013. Smarthome Configuration Model. In *Knowledge-based Configuration – From Research to Business Cases*, A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen (Eds.). Morgan Kaufmann Publishers, Chapter 10, 157–174. (Cited on page 136.)

X. Hu, X. Meng, and L. Wang. 2011. SVD-based Group Recommendation Approaches: An Experimental Study of Moviepilot. In *ACM Recommender Systems 2011 Challenge on Context-aware Movie Recommendation*. 23–28. (Cited on page 46.)

K. Jacowith and D. Kahneman. 1995. Measures of Anchoring in Estimation Tasks. *Personality and Social Psychology Bulletin* 21, 11 (1995), 1161–1166. (Cited on pages 4 and 66.)

A. Jameson. 2004. More than the Sum of its Members: Challenges for Group Recommender Systems. In *International Working Conference on Advanced Visual Interfaces*. 48–54. (Cited on pages 4, 20, 34, 49, 57, 58, 65, and 74.)

A. Jameson, S. Baldes, and T. Kleinbauer. 2004. Two Methods for Enhancing Mutual Awareness in a Group Recommender System. In *ACM Intl. Working Conference on Advanced Visual Interfaces*. Gallipoli, Italy, 447–449. (Cited on page 20.)

A. Jameson and B. Smyth. 2007. Recommendation to Groups. In *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.). Lecture Notes in Computer Science, Vol. 4321. 596–627. (Cited on pages 3, 4, 19, 21, 49, 50, 57, 58, and 152.)

A. Jameson, M. Willemsen, A. Felfernig, M. de Gemmis, P. Lops, G. Semeraro, and L. Chen. 2015. Human Decision Making and Recommender Systems. In *Recommender Systems Handbook, 2nd Edition*, F. Ricci, L. Rokach, and B. Shapira (Eds.). Springer, 611–648. (Cited on pages 20, 49, and 58.)

I. Janis. 1972. *Victims of groupthink: A psychological study of foreign decisions and fiascoes*. Houghton-Mifflin. (Cited on pages 4, 7, 8, and 67.)

D. Jannach, M. Zanker, A. Felfernig, and G. G. Friedrich. 2010. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA. (Cited on pages 1, 2, 3, 49, 57, 59, 65, 73, 75, 83, 84, 90, and 97.)

T. Joachims. 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 137–142. (Cited on page 116.)

U. Junker. 2004. QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems, In Proceedings of the 19th National Conference on Artifical Intelligence. *AAAI*, 167–172. (Cited on pages 87, 89, and 100.)

G. E. Kersten. 1997. Support for Group Decisions and Negotiations An Overview *. In *Multicriteria Analysis*, J. Clímaco (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 332–346. (Cited on page 143.)

R. Kilmann and K. Thomas. 1977. Developing a Forced-Choice Measure of Conflict-Handling Behavior: The "MODE" Instrument. *Educational and Psych. Measurement* 37, 2 (1977), 309–325. (Cited on page 143.)

M. Kompan and M. Bielikova. 2014. Group Recommendations: Survey and Perspectives. *Computing and Informatics* 33, 2 (2014), 446–476. (Cited on page 21.)

J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. 1997. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM* 40, 3 (1997), 77–87. (Cited on pages 2, 25, 83, and 97.)

Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37. (Cited on pages 45 and 46.)

D. Kudenko, M. Bauer, and D. Dengler. 2003. Group Decision Making through Mediated Discussions. In *User Modeling 2003*, P. Brusilovsky, A. Corbett, and F. de Rosis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 238–247. (Cited on page 58.)

H. Lamm and C. Sauer. 1974. Discussion-induced shift toward higher demands in negotiation. *European Journal of Social Psychology* 4, 1 (1974), 85–88. (Cited on pages 66 and 71.)

D. Leffingwell. 1997. Calculating the return on investment from more effective requirments management. 10 (1997), 13–16. (Cited on page 109.)

G. Leitner, A. Fercher, A. Felfernig, and M. Hitz. 2012. Reducing the Entry Threshold of AAL Systems: Preliminary Results from Casa Vecchia. In *13th Intl. Conference on Computers Helping People with Special Needs*. Linz, Austria, 709–715. (Cited on pages 2 and 84.)

G. Leitner, A. Fercher, A. Felfernig, K. Isak, S. Polat-Erdeniz, A. Akcay, and M. Jeran. 2016. Recommending and Configuring Smart Home Installations. In *Workshop on Configuration*. 17–22. (Cited on page 126.)

J. Levin and B. Nalebuff. 1995. An Introduction to Vote-Counting Schemes. *Journal of Economic Perspectives* 9, 1 (1995), 3–26. (Cited on pages 24 and 149.)

C. Li and Z. Luo. 2011. Detection of shilling attacks in collaborative filtering recommender systems. (10 2011). (Cited on page 8.)

J. Li, R. Jeffery, K. H. Fung, L. Zhu, Q. Wang, H. Zhang, and X. Xu. 2012. A Business Process-Driven Approach for Requirements Dependency Analysis. In *Business Process Management*, A. Barros, A. Gal, and E. Kindler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 200–215. (Cited on pages 5 and 110.)

S. L. Lim, D. Quercia, and A. Finkelstein. 2010. StakeNet: Using Social Networks to Analyse the Stakeholders of Large-scale Software Projects. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1 (ICSE '10)*. ACM, New York, NY, USA, 295–304. (Cited on page 5.)

E. Lind, L. Kray, and L. Thompson. 2001. Primacy Effects in Justice Judgments: Testing Predictions from Fairness Heuristic Theory. *Organizational behavior and human decision processes* 85 (08 2001), 189–210. (Cited on page 3.)

G. Linden, B. Smith, and J. York. 2003. Amazon.com Recommendations – Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. (Cited on page 25.)

J. Liu, P. Dolan, and E. R. Pedersen. 2010. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*. 31–40. (Cited on page 1.)

S. Loh, F. Lorenzi, R. Saldaña, and D. Licthnow. 2003. A tourism recommender system based on collaboration and text analysis. *Information Technology & Tourism* 6, 3 (2003), 157–165. (Cited on page 1.)

G. Macher, M. Atas, E. Armengaud, and C. Kreiner. 2015. Automotive Real-time Operating Systems: A Model-based Configuration Approach. *SIGBED Rev.* 11, 4 (Jan. 2015), 67–72. (Cited on page 6.)

T. Mahmood and F. Ricci. 2009. Improving Recommender Systems with Adaptive Conversational Strategies. In *20th ACM Conference on Hypertext and Hypermedia*. Torino, Italy, 73–82. (Cited on pages 20 and 21.)

M. Mandl, A. Felfernig, E. Teppan, and M. Schubert. 2011. Consumer Decision Making in Knowledge-based Recommendation. *J. Intell. Inf. Syst.* 37, 1 (Aug. 2011), 1–22. (Cited on page 65.)

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. (Cited on page 112.)

J. Marquez and J. Ziegler. 2015. Preference Elicitation and Negotiation in a Group Recommender Systems. In *Interact 2015 (LNCS)*, Vol. 9297. Springer, 20–37. (Cited on page 21.)

J. Masthoff. 2004. Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. *User Modeling and User-Adapted Interaction (UMUAI)* 14, 1 (2004), 37–85. (Cited on pages 22, 23, 24, 49, 57, and 74.)

J. Masthoff. 2006. The user as wizard: A method for early involvement in the design and evaluation of adaptive systems. In *Fifth Workshop on User-Centred Design and Evaluation of Adaptive Systems*, S. Weibelzahl, A. Paramythis, and J. Masthoff (Eds.). 460–469. Proceedings of the Fifth Workshop on User-Centred Design and Evaluation of Adaptive Systems, held in conjunction with the 4th International Conference on Adaptive Hypermedia & Adaptive Web-based Systems (AH'06), Dublin, Ireland, June 20th, 2006. (Cited on page 52.)

J. Masthoff. 2011. Group Recommender Systems: Combining Individual Models. *Recommender Systems Handbook* (2011), 677–702. (Cited on pages 2, 3, 4, 19, 20, 22, 24, 49, 50, 57, 58, 59, 63, 65, 73, 83, 92, 137, and 149.)

J. Masthoff. 2015. Group Recommender Systems: Aggregation, Satisfaction and Group Attributes. *Recommender Systems Handbook* (2015), 743–776. (Cited on pages 19, 20, 22, 24, and 149.)

J. Masthoff and A. Gatt. 2006. In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. *User Modeling and User-Adapted Interaction* 16, 3 (01 Sep 2006), 281–319. (Cited on page 58.)

K. McCarthy, L. McGinty, B. Smyth, and M. Salamó. 2006. Social Interaction in the CATS Group Recommender. In *Workshop on the Social Navigation and Community based Adaptation Technologies*. 743–776. (Cited on pages 20, 21, 39, 40, 41, 49, 58, and 74.)

D. McDonald and M. Ackerman. 2000. Expertise Recommender: A Flexible Recommendation System and Architecture. In *Conference on Computer Support Cooperative Work*. Philadelphia, PA, USA, 231–240. (Cited on page 47.)

L. McGinty and B. Smyth. 2003. On the Role of Diversity in Conversational Recommender Systems. In *Case-Based Reasoning Research and Development*, K. D. Ashley and D. G. Bridge (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 276–290. (Cited on page 59.)

S. M. McNee, S. K. Lam, J. A. Konstan, and J. Riedl. 2003. Interfaces for Eliciting New User Preferences in Recommender Systems. In *Proceedings of the 9th International Conference on User Modeling (UM03)*. Springer-Verlag, Berlin, Heidelberg, 178187. (Cited on page 141.)

D. McSherry. 2003. Similarity and Compromise. In *Case-Based Reasoning Research and Development*, K. D. Ashley and D. G. Bridge (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 291–305. (Cited on page 101.)

D. McSherry. 2004. Maximally Successful Relaxations of Unsuccessful Queries. In *15th Conference on AI and Cognitive Science*. AAAI Press, 127–136. (Cited on pages 29, 40, and 101.)

R. Van Meteren and M. Van Someren. 2000. Using content-based filtering for recommendation. In *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. 47–56. (Cited on page 2.)

V. Metsis, I. Androutsopoulos, and G. Paliouras. 2006. Spam filtering with naive bayes-which naive bayes?. In *CEAS*, Vol. 17. 28–69. (Cited on pages 116 and 119.)

B. Mobasher and J. Cleland-Huang. 2011. Recommender Systems in Requirements Engineering. *AI Magazine* 32, 3 (2011), 81–89. (Cited on pages 5, 109, and 110.)

A. Mojzisch and S. Schulz-Hardt. 2010. Knowing Others' Preferences Degrades the Quality of Group Decisions. *Journal of personality and social psychology* 98 (05 2010), 794–808. (Cited on pages 58 and 62.)

R. J. Mooney and L. Roy. 2000. Content-based Book Recommending Using Learning for Text Categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL '00)*. ACM, New York, NY, USA, 195–204. (Cited on pages 2 and 83.)

J. Murphy, C. Hofacker, and R. Mizerski. 2006. Primacy and Recency Effects on Clicking Behavior. *Journal of Computer-Mediated Communication* 11, 2 (2006), 522–535. (Cited on pages 66 and 111.)

D. G. Myers and H. Lamm. 1976. The group polarization phenomenon. *Psychological bulletin* 83, 4 (1976), 602. (Cited on pages 4, 66, and 68.)

V. Myllärniemi, J. Tiihonen, M. Raatikainen, and A. Felfernig. 2014. Using Answer Set Programming for Feature Model Representation and Configuration. In *Workshop on Configuration*. 1–8. (Cited on page 124.)

J. Neidhardt, L. Seyfang, R. Schuster, and H. Werthner. 2015. A Picture-based Approach to Recommender Systems. *Information Technology & Tourism* 15, 1 (2015), 49–69. (Cited on page 143.)

T. Nguyen. 2017. Conversational Group Recommender Systems. In *International Conference on User Modelling, Adaptation and Personalization (UMAP'17)*. ACM, 331–334. (Cited on page 20.)

T. Nguyen and F. Ricci. 2017. A Chat-Based Group Recommender System for Tourism. In *Information and Comm. Tech. in Tourism*, R. Schegg and B. Stangl (Ed.). 17–30. (Cited on page 20.)

G. Ninaus, A. Felfernig, M. Stettinger, S. Reiterer, G. Leitner, L. Weninger, and W. Schanil. 2014a. INTELLIREQ: Intelligent Techniques for Software Requirements Engineering. In *Prestigious Applications of Intelligent Systems Conference (PAIS)*. 1161–1166. (Cited on pages 5, 49, 58, and 74.)

G. Ninaus, F. Reinfrank, M. Stettinger, and A. Felfernig. 2014b. Content-based recommendation techniques for requirements engineering. In *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*. 27–34. (Cited on page 110.)

M. O'Connor, D. Cosley, J. Konstan, and J. Riedl. 2001. PolyLens: A Recommender System for Groups of Users. In *Europ. Conf. on Computer-Supported Cooperative Work*. ACM, 199–218. (Cited on pages 49, 58, and 73.)

F. Ortega, A. Hernando, J. Bobadilla, and J. H. Kang. 2016. Recommending Items to Group of Users using Matrix Factorization based Collaborative Filtering. *Information Sciences* 345, C (2016), 313–324. (Cited on pages 25, 45, and 46.)

D. Paraschakis. 2016. Recommender Systems from an Industrial and Ethical Perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 463–466. (Cited on page 97.)

J. Payne, J. Bettman, and E. Johnson. 1993. *The Adaptive Decision Maker*. Cambridge University Press, New York. (Cited on page 50.)

M. Pazzani and D. Billsus. 1997. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning* 27, 3 (01 Jun 1997), 313–331. (Cited on pages 2, 83, and 97.)

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830. (Cited on page 115.)

B. Peischl, M. Nica, M. Zanker, and W. Schmid. 2009. Recommending Effort Estimation Methods for Software Project Management. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 3. 77–80. (Cited on pages 2 and 84.)

D. Pennock, E. Horvitz, and C. Giles. 2000. Social Choice Theory and Recommender Systems: Analysis of the axiomatic foundations of collaborative filtering. In *17th National Conference on Artificial Intelligence (AAAI)*. Austin, TX, USA, 729–734. (Cited on page 19.)

R. Petty, J. Cacioppo, and D. Schumann. 1983. Central and Peripheral Routes to Advertising Effectiveness: The Moderating Role of Involvement. *Journal of Consumer Research* 10 (1983), 135–146. (Cited on page 50.)

J. Plisson, N. Lavrac, and D. Mladenic. 2004. A Rule based Approach to Word Lemmatization. In *Proceedings of the 7th International Multiconference Information Society. IS-2004*. 83–86. (Cited on page 113.)

C. Prud'homme, J. G. Fages, and X. Lorca. 2017. *Choco Documentation*. TASC - LS2N CNRS UMR 6241, COSLING S.A.S. (Cited on page 103.)

L. Quijano-Sánchez, D. Bridge, B. Díaz-Agudo, and J. Recio-García. 2013. A Case-Based Solution to the Cold-Start Problem in Group Recommenders. In *23rd International Conference on Artificial Intelligence (IJCAI 2013)*. 3042–3046. (Cited on page 25.)

L. Recalde. 2017. A Social Framework for Set Recommendation in Group Recommender Systems. In *European Conference on Information Retrieval*. Springer, 735–743. (Cited on page 47.)

R. Reiter. 1987. A Theory of Diagnosis from First Principles. *AI Journal* 32, 1 (1987), 57–95. (Cited on pages 9, 36, 37, 86, 87, 100, and 145.)

S. Reiterer. 2015. An Integrated Knowledge Engineering Environment for Constraint-based Recommender Systems. In *FINREC*. 11–18. (Cited on pages 2, 8, and 84.)

S. Reiterer, A. Felfernig, J. Michael, M. Stettinger, M. Wundara, and W. Eixelsberger. 2015. A Wiki-based Environment for Constraint-based Recommender Systems Applied in the E-Government Domain. In *UMAP Workshops*. 1–10. (Cited on pages 2, 3, 84, and 98.)

P. Resnick and H. R. Varian. 1997. Recommender Systems. *Commun. ACM* 40, 3 (March 1997), 56–58. (Cited on pages 6 and 65.)

F. Ricci, L. Rokach, and B. Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35. (Cited on pages 1, 57, 59, 65, and 97.)

F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. 2010. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg. (Cited on pages 1, 2, and 3.)

B. Van Roy and X. Yan. 2009. Manipulation-Resistant Collaborative Filtering Systems. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys 09)*. Association for Computing Machinery, New York, NY, USA, 165172. (Cited on page 8.)

K. Ryan. 1993. The role of natural language in requirements engineering. In *[1993] Proceedings of the IEEE International Symposium on Requirements Engineering*. 240–242. (Cited on page 110.)

D. Sacharidis. 2017. Group Recommendations by Learning Rating Behavior. In *International Conference on User Modelling, Adaptation and Personalization (UMAP'17)*. ACM, 174–182. (Cited on pages 22 and 25.)

M. Salamo, K. McCarthy, and B. Smyth. 2012. Generating Recommendations for Consensus Negotiation in Group Personalization Services. *Personal and Ubiquitous Computing* 16, 5 (2012), 597–610. (Cited on page 143.)

N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. Briand. 2017. An Automated Framework for Detection and Resolution of Cross References in Legal Texts. *Requir. Eng.* 22, 2 (June 2017), 215–237. (Cited on page 110.)

S. Schulz-Hardt, F. Brodbeck, A. Mojzisch, R. Kerschreiter, and D. Frey. 2006. Group Decision Making in Hidden Profile Situations: Dissent as a Facilitator of Decision Quality. *Journal of Personality and Social Psychology* 91, 6 (2006), 1080–1093. (Cited on pages 58 and 63.)

C. Schwind and J. Buder. 2012. Reducing confirmation bias and evaluation bias: When are preference-inconsistent recommendations effective - And when not? 28 (11 2012), 22802290. (Cited on page 72.)

C. Senot, D. Kostadinov, M. Bouzid, G. Picault, A. Aghasaryan, and C. Bernier. 2010. Analysis of Strategies for Building Group Profiles. In *Conference on User Modeling, Adaptation, and Personalization (UMAP 2010) (LNCS)*, Vol. 6075. Big Island, Hawaii, USA, 40–51. (Cited on pages 22, 24, and 149.)

C. Senot, D. Kostadinov, M. Bouzid, J. Picault, and A. Aghasaryan. 2011. Evaluation of Group Profiling Strategies. In *IJCAI 2011*. 2728–2733. (Cited on pages 22, 23, and 24.)

K. M. Shchekotykhin. 2014. Interactive Debugging of ASP Programs. *CoRR* abs/1403.5142 (2014), 1597–1603. (Cited on page 136.)

C. L. Sia, B. C. Y. Tan, and K. K. Wei. 2002. Group Polarization and Computer-Mediated Communication: Effects of Communication Cues, Social Presence, and Anonymity. *Information Systems Research* 13, 1 (2002), 70–90. (Cited on pages 66 and 71.)

H. Simon. 1955. A Behavioral Model of Rational Choice. *Quarterly Journal of Economics* 69 (1955), 99–118. (Cited on page 50.)

T. Soininen and I. Niemelä. 1998. Developing a declarative rule language for applications in product configuration. In *PADL*. 305–319. (Cited on pages 124 and 140.)

G. Stasser and W. Titus. 1985. Pooling of unshared information in group decision making: Biased information sampling during discussion. *Journal of personality and social psychology* 48, 6 (1985), 1467. (Cited on pages 3, 7, 12, 62, and 63.)

M. Stettinger and A. Felfernig. 2014. CHOICLA: Intelligent Decision Support for Groups of Users in Context of Personnel Decisions. In *ACM RecSys'2014 IntRS Workshop*. Foster City, CA, USA, 28–32. (Cited on pages 29 and 138.)

M. Stettinger, A. Felfernig, G. Leitner, and S. Reiterer. 2015a. Counteracting Anchoring Effects in Group Decision Making. In *23rd Conference on User Modeling, Adaptation, and Personalization (UMAP'15) (LNCS)*, Vol. 9146. Dublin, Ireland, 118–130. (Cited on pages 4, 8, 50, 58, and 61.)

M. Stettinger, A. Felfernig, G. Leitner, S. Reiterer, and J. Michael. 2015b. Counteracting Serial Position Effects in the CHOICLA Group Decision Support Environment. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 148–157. (Cited on pages 50, 58, 59, 60, 61, 66, 74, 111, and 145.)

M. Stolze, S. Field, and P. Kleijer. 2000. Combining Configuration and Evaluation Mechanisms to Support the Selection of Modular Insurance Products. In *Proceedings of the 8th European Conference on Information Systems, Trends in Information and Communication Systems for the 21st Century, ECIS 2000, Vienna, Austria, July 3-5, 2000*. 858–865. (Cited on page 6.)

M. Stone. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)* 36, 2 (1974), 111–147. (Cited on page 116.)

J. A. F. Stoner. 1961. *A comparison of individual and group decisions involving risk*. Ph.D. Dissertation. Massachusetts Institute of Technology. (Cited on pages 4, 8, 66, 68, and 71.)

M. Stumptner. 1997. An Overview of Knowledge&Dash;Based Configuration. *AI Commun.* 10, 2 (April 1997), 111–125. (Cited on pages 5, 10, and 123.)

C. R. Sunstein. 2002. The law of group polarization. *Journal of political philosophy* 10, 2 (2002), 175–195. (Cited on pages 4, 66, and 68.)

R. Taupe, A. Falkner, and G. Schenner. 2016. Deriving Tighter Component Cardinality Bounds for Product Configuration. In *18 th International Configuration Workshop*. 47. (Cited on page 136.)

E. Teppan and G. Friedrich. 2016. Heuristic Constraint Answer Set Programming. In *ECAI 2016*. 1692–1693. (Cited on pages 124, 132, and 136.)

J. Tiihonen, T. Soininen, I. Niemelä, and R. Sulonen. 2003. A practical tool for mass-customizing configurable products. In *14th International Conference on Engineering Design*. 1290–1299. (Cited on page 124.)

M. Tkali, B. Carolis, M. de Gemmis, A. Odic, and A. Kosir. 2016. *Emotions and Personality in Personalized Services: Models, Evaluation and Applications*. (Cited on page 143.)

O. Torp and O. Klakegg. 2016. Challenges in Cost Estimation under UncertaintyA Case Study of the Decommissioning of Barsebck Nuclear Power Plant. 6 (10 2016), 14. (Cited on pages 68 and 72.)

M. Torrens, P. Hertzog, L. Samson, and B. Faltings. 2003. Reality: A Scalable Intelligent Travel Planner. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC '03)*. ACM, New York, NY, USA, 623–630. (Cited on pages 2, 8, and 84.)

O. T. Tran, B. X. Ngo, M. L. Nguyen, and A. Shimazu. 2014. Automated reference resolution in legal texts. *Artificial Intelligence and Law* 22, 1 (01 Mar 2014), 29–60. (Cited on page 110.)

B. Trstenjak, S. Mikac, and D. Donko. 2014. KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering* 69 (2014), 1356 – 1364. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013. (Cited on page 116.)

E. P. K. Tsang. 1993. *Foundations of constraint satisfaction*. Academic Press. I–XVIII, 1–421 pages. (Cited on pages 85, 99, and 124.)

B. W. Tuckman. 1965. Developmental sequence in small groups. *Psychological bulletin* 63, 6 (1965), 384. (Cited on page 142.)

B. W. Tuckman and M. A. C. Jensen. 1977. Stages of small-group development revisited. *Group & Organization Studies* 2, 4 (1977), 419–427. (Cited on page 142.)

A. Tversky and I. Simonson. 1993. Context-dependent Preferences. *Manage. Sci.* 39, 10 (Oct. 1993), 1179–1189. (Cited on page 66.)

D. Čubranić. 2004. Automatic bug triage using text categorization. In *In SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. KSI Press, 92–97. (Cited on page 118.)

Y. Wang, W. Dai, and Y. Yuan. 2008. Website browsing aid: A navigation graph-based recommendation system. *Decision Support Systems* 45, 3 (2008), 387–400. (Cited on page 1.)

A. Weinzierl. 2017. Blending Lazy-Grounding and CDNL Search for Answer-Set Solving. 191–204. (Cited on page 136.)

Glen Whyte. 1989. Groupthink reconsidered. *Academy of Management Review* 14, 1 (1989), 40–56. (Cited on page 4.)

D. R. Wilson and T. R. Martinez. 1997. Improved Heterogeneous Distance Functions. *J. Artif. Int. Res.* 6, 1 (Jan. 1997), 1–34. (Cited on page 102.)

K. Win and B. Srisura. 2019. *Approaching Mobile Constraint-Based Recommendation to Car Parking System*. 306–313. (Cited on pages 2 and 84.)

D. Winterfeldt and W. Edwards. 1986. *Decision Analysis and Behavioral Research*. Cambridge University Press. (Cited on pages 33, 74, and 100.)

G. M. Wittenbaum, A. B. Hollingshead, and I. C. M. Botero. 2004. From cooperative to motivated information sharing in groups: moving beyond the hidden profile paradigm. 286–310. (Cited on pages 3, 58, and 62.)

W. Wobcke, A. Krzywicki, Y. Kim, X. Cai, M. Bain, P. Compton, and A. Mahidadia. 2015. A Deployed People-to-People Recommender System in Online Dating. *AI Magazine* 36, 3 (2015), 5–18. (Cited on page 47.)

H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok. 2008. Interpreting TF-IDF Term Weights As Making Relevance Decisions. *ACM Trans. Inf. Syst.* 26, 3 (June 2008), 13:1–13:37. (Cited on page 115.)

L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *ACM Conference on Recommender Systems (RecSys'17)*. ACM, 107–115. (Cited on page 143.)

I. Yaniv. 2011. Group diversity and decision quality: Amplification and attenuation of the framing effect. *International Journal of Forecasting* 27, 1 (2011), 41 – 49. (Cited on pages 7 and 58.)

Z. Yu, X. Zhou, Y. Hao, and J. Gu. 2006. TV Program Recommendation for Multiple Viewers based on User Profile Merging. *User Modeling and User-Adapted Interaction* 16, 1 (2006), 63–82. (Cited on pages 21 and 33.)

X. Yuan, J. H. Lee, S. J. Kim, and Y. H. Kim. 2013. Toward a user-oriented recommendation system for real estate websites. *Information Systems* 38, 2 (2013), 231–243. (Cited on page 1.)

Bingsheng Zhang and Hong-sheng Zhou. 2017. Brief announcement: Statement voting and liquid democracy. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. 359–361. (Cited on page 142.)

D. Zhang, P. B. Lowry, L. Zhou, and X. Fu. 2007. The Impact of IndividualismCollectivism, Social Presence, and Group Diversity on Group Decision Making Under Majority Influence. *Journal of Management Information Systems* 23, 4 (2007), 53–80. (Cited on page 142.)