Technische Universität München
Forschungs- und Lehreinheit XI
Angewandte Informatik / Kooperative Systeme

Improving the User Experience in Mobile Recommender Systems

Béatrice Lamche

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. J. Ott
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. J. Schlichter
	2.	Univ.-Prof. Dr. A. Butz (Ludwig-Maximilians-Universität München)

Die Dissertation wurde am 05.11.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 11.12.2015 angenommen.

Abstract

Mobile recommender systems support the decision making process of users by providing suggestions for items that are of potential use for them in a certain mobile context. However, due to smaller display sizes in mobile interfaces, reduced attention span and uncertainty of the user's preferences in the beginning, the improvement of the user experience remains one of the main challenges when designing mobile recommender systems and has not been investigated thoroughly. This dissertation first identifies important characteristics that lead to a user-friendly (mobile) recommender system. Based on this analysis, a framework is conceptualized which consists of five building blocks that require further research: Active Learning, interactive explanations, presentation, context and user modeling. Each building block of the framework is evaluated from a user's perspective within sub-projects and then combined in a final evaluation. A key contribution lies in the evaluation of this framework consisting of concrete presentation guidelines, efficient algorithms that take Active Learning as well as mobile context into account, a strategy to automatically generate interactive explanations and a suitable stereotype user model, to indicate how developers can improve the user experience of mobile recommender systems.

Kurzfassung

Mobile Empfehlungssysteme unterstützen Nutzer durch Vorschläge, die im jeweiligen mobilen Kontext bei der Entscheidungsfindung hilfreich sein können. Durch geringe Displaygrößen, reduzierte Aufmerksamkeitsspanne und zu Beginn undefinierten Nutzerpräferenzen, ist die Erzeugung eines positiven Nutzererlebnisses eine der größten Herausforderungen bei der Entwicklung dieser Systeme. Im Rahmen der Dissertation werden deshalb zunächst jene Charakteristika identifiziert, die ein nutzerfreundliches (mobiles) Empfehlungssystem ausmachen. Basierend darauf wird ein Framework entwickelt, welches aus fünf Bausteinen besteht, die es im Hinblick auf mobile Empfehlungssysteme noch zu untersuchen gilt: Aktives Lernen, interaktive Erklärungen, Darstellung, Kontext und Nutzermodellierung. Jeder dieser Bausteine wird erst aus der Nutzerperspektive heraus innerhalb von Teilprojekten und anschließend in einer allumfassenden Nutzerstudie evaluiert. Ein wichtiger Beitrag liegt in der Evaluierung dieses Frameworks, bestehend aus konkreten Darstellungs-Richtlinien, effizienten Algorithmen, die aktives Lernen und mobilen Kontext mit einbeziehen, einer Methodik zur automatischen Erzeugung mobiler Erklärungen und schließlich einem geeigneten Stereotypen-basierten Nutzermodell, um Entwickler von mobilen Empfehlungssystemen dabei zu unterstützen, ein positives Nutzererlebnis zu erzeugen.

Acknowledgments

This work would not have been possible without the guidance, encouragement and inspiration of my colleagues, friends and family.

First and foremost I would like to express my deepest gratitude to my advisor, Prof. Dr. Johann Schlichter, for giving me the opportunity to join his research group and for his guidance, caring and patience. I also would like to thank my second advisor, Prof. Dr. Andreas Butz, for providing me with valuable feedback and giving final shape to this thesis.

I had the privilege of working with an incredible group of colleagues at the department of *Applied Informatics - Cooperative Systems*. Lots of valuable ideas were created based on interesting discussions with the whole team. I really enjoyed my time as a PhD student, mostly because of the great atmosphere and team spirit. In particular, I am grateful to Georg, Florian, Claudius, Niklas, Alexander, Michele, Friedl, Marlene, Karim, Hubert, Christoph, Matthias and Marouane for collaborating with me and cheering me up with kind words and support. A special thanks goes hereby to Wolfgang. He was a mentor to me, always willing to help and give his best suggestions. Whenever I had questions, he gave advice and provided me with significant resources and insights into the field of recommender systems from the very beginning. I am also grateful to all the individuals that have donated their time as participants in my user studies and all the students who contributed to this dissertation.

I would also like to thank my friends for many years of adventures and great moments spent together. I am truly blessed to know all of them and although the majority did not really understand what I was doing, I would not have made it without their moral support. My sincere appreciation goes to my boyfriend Chris for always believing in me and constant encouragement, even from many miles away. To my sister Iris, and my parents Jasmine and Werner: Words cannot express the depth of gratitude I feel for a lifetime love and support.

THANK YOU!

Contents

1

Introduction

Mobile phones are more and more used for information access. While earlier mobile phones provide only limited functionality, e.g., phone calls and short messaging services, current smartphones can be customized by choosing from a variety of applications, also called *apps* [Böhmer et al., 2013]. In February 2014, there have been 1.3 million applications available for Android phones in the *Google Play Store* and, as of July 2014, 1.2 million applications for iPhones in the *Apple App Store* [Statista, 2015]. The number of available applications is constantly increasing. Due to corresponding technological developments in this area, such as the ubiquitous availability of wireless communication services and position detection techniques, the amount of information and web services increases. Hence, it becomes more and more difficult for mobile users to filter the necessary information to complete a specific task. Recommendation techniques are widely used to improve the usability of mobile systems and to help deal with this information overload by providing personalized suggestions based on some recommendation algorithms. The recommendations relate to various decision-making processes, such as which product to buy, which restaurant to choose or which movie to watch. What makes a recommender system successful is its interaction design, graphical user interface and recommendation algorithm. The task of these systems is not only to accurately suggest items, but also to improve user satisfaction and to gain knowledge about the users' preferences [Ricci et al., 2011]. Although lots of research has focused on improving the accuracy of web-based recommender systems, the user experience is getting more and more important nowadays [Konstan and Riedl, 2012].

> *"Successful mobile products are ones that are useful and usable, and provide a coherent, comprehensive user experience [Jones and Marsden, 2006, p.39]".*

The term "*user experience*" (UX) became a buzzword in the field of human-computer interaction (HCI) and is associated with different meanings and interpretations. Although sometimes user experience is just used as a synonym for usability, academics emphasize the difference between both terms [Hassenzahl, 2008]. When designing interactive products, there are certain usability, as well as user experience goals to be aimed for. Usability

goals aim at meeting specific usability criteria, e.g., efficiency or effectiveness, whereas, user experience goals are concerned with explaining the nature of the user experience, e.g., to provide a pleasing and aesthetic satisfaction while interacting with the system. However, it is not possible to draw a distinct line between these two types of goals, since usability is often essential to the quality of the user experience and, conversely, aspects of the user experience, such as how it feels and looks, are linked with the product's usability. In the past, human-computer interaction mostly focused on usability issues (known as usability engineering) but has nowadays become concerned with understanding, evaluating and improving user experience aspects [Rogers et al., 2011]. Hassenzahl [Hassenzahl, 2008] defines user experience as a:

> *"momentary, primarily evaluative feeling (good-bad) while interacting with a product or service [Hassenzahl, 2008, p.2]".*

In other words, user experience shifts focus from the product and its functions to users' feelings when using the product. This is a dynamic phenomenon that changes over time. However, the question remains how a positive user experience can be designed. Based on the consideration of certain psychological needs, Hassenzahl distinguishes between three different dimensions along which interactive products can be perceived. The first one, *pragmatic quality*, refers to the product's ability to help achieving certain *do goals*. It describes the *what* of the user's activity, e.g., making a phone call. The second dimension, *hedonic quality*, refers to the user's *be goals*. The focus hereby is on the *self* and describes the motivation *why* someone is using a certain product. In the telephone example, calling someone can spark an emotional experience, e.g., when calling a close friend [Hassenzahl, 2008]. In a subsequent work, Hassenzahl adds a third goal that influences the user's perception of the product: The *motor goal*. This goal addresses *how* a product is used (e.g., initiating a call by pressing buttons). When designing interactive products, attention basically lies on *do-* and *motor goals*. However, Hassenzahl argues that the fulfillment of *be goals* is the actual driver of experience [Hassenzahl, 2010]. *Figure 1.1* illustrates all three dimensions.

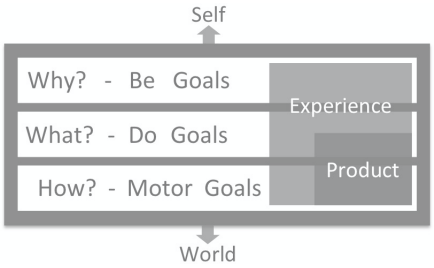


Figure 1.1: The user experience model adapted from [Hassenzahl, 2010]

Also developers of recommender systems recognized the importance of a product's ability to create positive experiences. For example developers of businesses applications nowadays not only address the accuracy of the recommendation algorithm, but more and more focus on the creation of a positive customer experience to better fit with their sales and marketing strategies. This also involves the integration of business logic, e.g., implementing specific rules to prevent recommending sold-out items. However, measuring the user experience of an interactive product is a challenging task. While the system's accuracy can be measured by using existing datasets, the user experience can only be

investigated by developing a prototype and conducting user studies [Konstan and Riedl, 2012]. These prototypes have to be developed in a certain way that allows the perception of the user experience in order to evaluate it in a user study. Moreover, engineers have to obtain a broad understanding of the users' psychological needs to design products with a positive user experience, such as the need for autonomy, competence or relatedness [Butz and Krüger, 2014]. Of course, prediction accuracy still plays a decisive role for the success of a recommender system, however studies have shown that increasing the system's accuracy does not automatically improve the user experience. The researchers [Swearingen and Sinha, 2001] found out that from a user's perspective, an effective recommender system is transparent, supports exploration of new items, offers detailed item descriptions including pictures and rating, as well as provides interaction methods to allow the user to refine the recommendations. Research has also shown that user experience is improved when the recommender system gives the user control over recommendations (e.g., by making the system's functionality transparent, offering interaction methods and taking the user's context into account) even if this control increases the workload on users and does not improve the accuracy of recommendations. The challenge for developers of (mobile) recommender systems is therefore to deliver accurate recommendations, while also creating a positive user experience [Konstan and Riedl, 2012].

1.1 Research Question

The above mentioned challenges for mobile recommender systems open a large research space regarding the improvement of usability, user experience and satisfaction of (mobile) recommender systems. This dissertation is intended to decrease this large research space by presenting solutions on how to improve the user experience of mobile recommender systems. A solution to this problem will lead to a higher acceptance of mobile recommender systems, valuable for businesses, as well as for the customer itself. A mobile system generating personalized shopping recommendations helps the user to find the most satisfying product by reducing search effort and information overload [Liang et al., 2006]. Also businesses will have interest in such a system which attracts more customers to the stores. Very diverse application scenarios can be imagined, such as restaurant, points of interest or leisure activity recommender systems. Persuasive recommender systems on the other hand aim at convincing the user to a healthier or more sustainable lifestyle in the long term. Application examples include mobile systems that recommend healthy recipes and ingredients that also match the user's preferences. Moreover, a system that selects sustainable products (e.g., fair trade groceries) or presents convenient energy-saving measures can be imagined. Here, mobile recommender systems make the user's life not only easier but also support a healthier or more sustainable lifestyle. Hence, any contribution in this direction must be considered as valuable.

Currently, research on recommender systems heavily focused on the development of accurate algorithms and mostly neglected the user experience. However, only user-friendly products will be accepted by the majority of users, no matter how accurate the algorithm

generated recommendations. Moreover, user studies investigating the user experience of recommender systems mostly focused on desktop devices. Simply overtaking the solutions for these systems which have a bigger screen and different interaction methods, would lead to frustration of the users and a negative user experience of the system when used on a mobile device.

In general, some of the aspects of non-mobile recommender systems that have the ability to improve the user experience can probably also be applied for mobile devices. However, they first need to be identified based on analyzing previous research and then evaluated in a prototype. New aspects such as mobile context and new interaction methods, specific for small touch screen devices, have to be considered as well. A conceptual framework which presents all characteristics of a mobile recommender system with a positive user experience remains a research gap. Based on the challenges and issues discussed above, we investigate the following question:

Overall Research Question: How can the user experience of mobile recommendation systems be improved by maintaining accuracy?

1.2 Research Methodology

In order to find out how to improve the user experience of mobile recommender systems, traditional human-computer interaction methods have to be applied. This dissertation project aims at developing a conceptual framework that describes all necessary characteristics that need to be considered when designing an efficient mobile recommender system from a human-computer interaction perspective. First, the domain of (mobile) recommender systems has to be comprehensively analyzed. Based on this analysis, success factors of mobile recommender systems with a positive user experience will be identified. These success factors create the foundation of the conceptual framework and act as "building blocks". The framework will therefore consist of several building blocks that have the potential to contribute to an enhanced user experience of mobile recommender systems. Each of these building blocks requires further research and will be evaluated separately, both theoretically and empirically. For each experiment, a prototype will be developed, so that the theoretical considerations can be evaluated within a user study. The outcome of each of the experiments will iteratively extend parts of the framework for mobile recommender systems with a positive user experience. A final evaluation, taking all identified success factors into account, will verify the applicability of the conceptual framework. As a result, an evaluated framework consisting of concrete presentation guidelines, efficient algorithms and suitable user modeling approaches will be proposed that supports developers to improve the user experience of mobile recommendations by maintaining accuracy.

In this spirit, the thesis follows a design science approach proposed by [von Alan et al., 2004]. It meets the seven guidelines for design science in information systems research.

G1: Creation of several concepts and prototypes of a mobile recommender system that support a positive user experience by generating user-friendly recommendations for a smartphone in a specific application scenario (*"Design as an Artifact"*).

G2: Demonstration of the relevance of the research question by presenting related work that discusses the need for improving the user experience of mobile recommender systems, both for business, as well as for research endeavors (*"Problem Relevance"*).

G3: Discussion of different evaluation metrics and data to evaluate the designed artifacts in an appropriate manner, both quantitatively and qualitatively (*"Design Evaluation"*).

G4: Valuable contributions in research areas of HCI and Mobile Recommender Systems in terms of developed concepts and prototypes, demonstrating an improved user experience of mobile recommender systems (*"Research Contributions"*). *Section 1.3* presents a detailed list of research contributions.

G5: Application of rigorous scientific methods derived from related work in this research field, to define the requirements of a mobile recommender system with a positive user experience and to iteratively construct and evaluate the prototypes (*"Research Rigor"*).

G6: Evaluation of developed artifacts to iteratively refine the requirements and improve the prototypes step by step (*"Design as a Search Process"*).

G7: Scientific presentation, publication and discussion of the developed and evaluated concepts and solutions in technology- and business-oriented international conferences and journals, e.g., [Lamche, 2014, Lamche et al., 2014b, Lamche et al., 2015b] (*"Communication of Research"*).

1.3 Contributions

Based on the research methodology presented in the previous section, this thesis provides several contributions to two subfields of computer science: Mobile Recommender Systems and HCI. This thesis aims to provide particularly the following contributions:

C1: Evaluation of different Active Learning strategies and concrete suggestions for their application depending on the specific scenario.

C2: Proposal of a suitable user model for mobile recommender systems based on stereotypes.

C3: Concrete guidelines for the design of interactive user interfaces for mobile recommender systems.

C4: Better understanding of mobile context and how it can be used to improve mobile recommendations.

C5: Detailed understanding of what explanations should look like and how they can be generated automatically.

Overall, the final evaluated conceptual framework should give mobile recommender system developers an understanding of how to generate user-friendly and accurate recommendations in mobile scenarios. With this knowledge, mobile recommender systems can be developed more quickly and successfully.

1.4 Thesis Structure

The section below presents the outline of the thesis and summarizes the research endeavors of each chapter briefly.

Chapter 1 defines the term "User Experience" and presents the motivation, the main research question, methodology as well as the contributions of this work.

Chapter 2 gives an overview of the fundamentals of (mobile) recommender systems. It first examines the term "recommender system" by presenting the task and history of recommender systems. Then the functionality of a recommendation process and its different approaches are explained. Since our solution is based on a critiquing-approach, the concept behind conversation-based recommender systems and its variants is described. Evaluation techniques of this user-centered design process are discussed in order to iteratively improve the prototypes and come up with solutions that support a positive user experience of a mobile recommender system. We also present a classification of mobile recommender systems and discuss existing approaches and relevant results in order to verify our research question.

Chapter 3 investigates which HCI aspects have to be considered when developing a mobile recommender system with a positive user experience. The user-friendliness plays an important role for the success of a recommender system, but also other characteristics have to be taken into account. Based on previous research we determine the most important aspects of mobile recommender system with a positive user experience and summarize them in a conceptual framework. The building blocks of this framework display the current research gaps regarding mobile recommender systems with a positive user experience and will be investigated in more detail in the following chapters. We conclude the chapter by presenting possible application scenarios and a justification of the selected main scenario.

Chapter 4 reflects the implementation and evaluation of a mobile shopping recommender system using Active Learning. It explains the reasoning for the selection of the

Active Learning algorithm and the details of the design process of the prototype – a mobile shopping recommender system that integrates a conversation-based Active Learning strategy. In a user study we test two variants of the system: an algorithm using a diversity-based and one using a similarity-based approach. The results of the user study show that the new diversity-based approach enhances the user experience and is preferred by the majority of users.

Chapter 5 discusses the cold-start problem in a mobile recommender system and presents a solution to generate a user model right from the beginning. The solution is based on stereotypes that promise to deliver fast and accurate personalized recommendations. A critiquing shopping application first uses navigation by asking to determine the user's stereotype to deliver personalized recommendations. Then, navigation by proposing supports critiquing in a finer granularity level to refine the user model. We conduct a user study that shows that a prototype using a stereotype-based user model performs better than a system without a stereotype-based logic.

Chapter 6 investigates how to design the interaction and usability of a mobile shopping recommender system. Smartphones reveal additional characteristics compared to desktop systems due to smaller screens and a direct touch input method and can in addition collect information about the current environment. Therefore, we conduct an interaction design process which involves work on establishing requirements for a mobile recommender system, designing solutions that meet these requirements and produce a low- and higher-fidelity prototype of the solution. The two-part evaluation allows for the specification of interaction design guidelines that help improving the user experience of a mobile recommender system.

Chapter 7 explores if the integration of context-aware information can improve mobile recommendations. We first assess the influence of different context factors on the shopping behavior by conducting a survey. Based on these results, a context-aware mobile shopping recommender system is developed in order to find out if contextual information such as weather, budget and shopping intent can predict the user's current shopping interest to improve accuracy, efficiency, as well as the user experience of a mobile shopping recommender system. Results of our evaluation show that a mobile shopping recommendation application that takes the user's context into account, performs better than an application that does not consider mobile context.

Chapter 8 focuses on explanations of mobile recommendations. Explanations of recommendations help users to make better decisions in contrast to recommendations without explanations while also increasing the transparency between the system and the user. We develop a strategy to generate interactive explanations for a content-based recommender system. Within our approach, the user is now allowed to change wrong assumptions made by the system by interacting with the explanations. A mobile application for a shopping scenario is developed and evaluated by following the proposed concept. Results show that

the user experience is improved by generating interactive explanations of mobile recommendations.

Chapter 9 deals with the overall final evaluation of the developed concepts. In the previous chapters, concepts and prototypes have been developed for each of the building blocks of the framework and evaluated separately. However, in order to validate the conceptual framework, a prototype implementing all of the previous findings and solutions is evaluated in an overall user study with one hundred participants. Results are very promising and prove that the conceptual framework helps improving the user experience of a mobile recommender system.

Chapter 10 concludes this thesis by presenting critical aspects and limitations. Moreover, we summarize the main contributions to support developers to improve the user experience of their mobile recommendation systems. The last section of this work indicates related research questions and improvements for future work.

Hereafter, to avoid the continuous repetition of "he or she", this thesis will use "she", as well as "her", representative for both genders referring to users, developers and other humans in the system.

2

Fundamentals of Mobile Recommender Systems

This chapter introduces the theoretical background and the relevant concepts that form the basis of this thesis. The first section (1) examines the term "Recommender System" by giving a short overview about the task and history of recommender systems. Next, it explains the functionality of a recommendation process and its different approaches. Section (2) introduces the concept behind conversation-based recommender systems and its variants. Since our main goal is to improve the user experience of mobile recommender systems, section (3) presents evaluation techniques of this user-centered design process. The last section of this chapter (4) establishes a classification of mobile recommender systems, distinguishes between different application tasks and presents existing systems. Based on the analysis of related work we also verify the research gap and determine our research question.

2.1 Recommender Systems

Research on recommender systems has developed during the mid-1990's. It is among others based on the observation that people place high value on recommendations provided by others when making decisions such as considering to buy a product of a certain type. The term recommender system is described by Robin Burke, one of the first researchers in this area, as follows:

> *". . . any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options." – [Burke, 2002]*

Recommender systems are tools and related techniques supporting the decision making process of users by providing suggestions for items that are of potential use for them. Examples for such decisions are: Which clothes to buy, what music to listen to, which restaurant to visit or which online news source to consult. A recommender system usually focuses on a specific type of item and all of its parts are designed in a way that specifically fits its purpose. A distinction can be made between personalized recommendations and non-personalized recommendations. The website *Amazon.com*, for example, uses personalized recommendations so that each user or user group receives different recommendations based on previous activity on the site. Typical examples for non-personalized recommendations are top ten lists based on user ratings or sales. Recommender system research focuses on personalized recommendations as non-personalized recommendations are easy to generate. Personalized recommendations are based on *user models* and exploit the information stored in them [Ricci et al., 2011].

The following section of this paper reviews a wide range of recommender systems research. It explains the characteristics and functionality of recommender systems in order to convey the basics of recommender systems necessary for developing mobile recommender systems. The first part of this section introduces the characteristics of recommender systems. It follows a detailed consideration of conversation-based recommender systems. The third part focuses on mobile recommender systems. It presents different classifications of mobile recommender systems to also give an overview of state-of-the-art approaches. Subsection four discusses evaluation approaches of (mobile) recommender systems.

2.1.1 Search Task Variants

Users of recommender systems can basically engage in two different kinds of search activities [Marchionini, 2006].

Look Up and Exploitation

Look up is a simple search task where the user enters a search query and receives discrete and well-structured objects such as names, text information, specific items or other media. This kind of search task is also supported by traditional web search engines [Marchionini, 2006]. In this scenario, the system exploits the user's clearly stated preferences in order to return appropriate recommendations. Regarding the user modeling approach, *exploitation* means that a user model already exists and the system uses it to predict ratings or make recommendations [Perugini et al., 2004]. The authors [Butz and Krüger, 2014] describe this search task variant simply as *searching*.

Exploratory Search

Within exploratory search tasks, users are unable to formulate an explicit query to solve their task. The user either might not have a specific item in mind and wants to explore the search space or cannot anticipate which keywords to type in [Stewart et al., 2008]. Regarding mobile devices, limited resources regarding input capabilities, smaller display sizes and other restrictions of small mobile devices might be another reason why the user cannot provide a clear search query. Recommender systems supporting exploratory search actively engage the user in the search process, e.g., by asking the user to rate products [Marchionini, 2006,Perugini et al., 2004]. Exploration also affects the user modeling process of a recommender system. A system enables exploration in order to learn more about the user's preferences. This is not only important to construct a user model for a new user about whom nothing or little is known, but also for a returning user who wishes to get inspired, see new and different things and expand the knowledge of the item space [Rubens et al., 2011]. This search task variant can also be defined as *browsing* [Butz and Krüger, 2014].

2.1.2 Knowledge Sources

One of the first steps when building a recommender system is to decide on a knowledge source the calculation of recommendation is drawn on. Recommender systems can base their consumptions on different forms of knowledge. The authors Felfernig and Burke distinguish between three knowledge sources that can be used [Felfernig and Burke, 2008]. First, the user herself. This includes the user's preferences, demographics and queries. Second, other peer users of the system can be considered. Here, recommendations are based on peer demographics and opinions. Third, recommendations can be generated by taking item attributes and knowledge about how recommended items are used into account [Felfernig and Burke, 2008].

2.1.3 User-System Lifecycle

Building a user model to predict its preferences is a challenging task since the user's needs and interests may change over time. The authors Konstan and Riedl define three stages where a recommender system must understand the different needs of the users and act appropriately [Konstan and Riedl, 2012].

New Users

Recommender systems build up a user model in order to deliver personalized recommendations. The "new user problem" describes the challenge to build up a user model of a first-time user. There are basically two approaches to build the user model. On the one

hand, the system can present different items to the user, e.g., based on demographic information or most-popular items. On the other, the system can use specific algorithms to present items to the user to elicit ratings [Konstan and Riedl, 2012]. This strategy is also called *Active Learning* and will be discussed in more detail in *chapter 4*.

Changing Preferences

In the beginning of the recommendation process, it is necessary to ask for more user ratings to narrow down the interests. When a rich user model has already been built up, the rating contribution can be reduced, however a recommender system has to consider that the user's interests may change over time [Konstan and Riedl, 2012]. Distinguishing between long-term and short-term user preferences is one approach to handle this problem. Some of the user's characteristics may stay stable (e.g., date of birth, gender or nationality). They have only to be elicited once. Specific interests and needs can change according to the context. Therefore, the recommender system should not focus too much on a specific item space and adapt the user's preferences (e.g., in form of a short-time model) from time to time [Ricci et al., 2011]. *Chapter 5* will describe a suitable approach to tackle this challenge.

New Items

Recommender systems not only serve the user's individual benefits but often also the community as a whole. Ratings of products are necessary both for a more accurate user profile, but also for generating better recommendations for other users. Especially when a new item is introduced to the system, the system has to make a decision to whom it should be recommended. It can either try to obtain the item's characteristics, e.g., by cooperations with experts or image recognition algorithms, or by asking users to rate the item. The challenge hereby is to not bother the users and to demonstrate how these contributions help the whole community [Konstan and Riedl, 2012].

2.1.4 Traditional Recommendation Techniques

Another decision that has to be made when developing a recommender system is which recommendation algorithm should be used to process the data of the knowledge source. Several types of recommendation techniques exist, however, different authors suggest different categorizations. While the authors Adomavicius and Smyth mention three types of techniques (*collaborative filtering, content-based-* and *hybrid techniques* [Adomavicius and Tuzhilin, 2005,Smyth, 2007]), Burke suggests five different types [Burke, 2002]. Ricci takes it a step further by arguing six different types of recommendation algorithms [Ricci et al., 2011]. The most common recommendation techniques are discussed in the following:

Collaborative Filtering

This recommendation algorithm is probably the most widely implemented one. Systems employing this technique help users to make decisions based on ratings of other users. Items that similar users liked in the past are recommended rather than relying on the actual properties of items. These systems assume that users with similar taste rate items similarly. Therefore, they rely on the availability of user ratings on items in order to make useful recommendations [Burke, 2002].

Content-Based

Recommender systems using a content-based recommendation algorithm suggest the user items similar to the ones the user preferred in the past. These systems assume that users rate items with similar features similarly [Ricci et al., 2011]. One special case of content-based recommender systems are *case-based* recommender systems which depend on the structural representation of items with a well-defined set of properties. For instance, in a clothing items recommendation scenario, items could be presented with features such as price, color, brand, type, and so on. Such systems use these features and similarity knowledge to make judgments on how similar items are to one another and to a query [Smyth, 2007].

Knowledge- and Utility-Based

Knowledge- and utility-based recommender systems suggest items based on inferences about a user's preferences by making use of functional knowledge about how certain product features meet user needs. Any knowledge structure that supports such an inference can be used as a user profile [Burke, 2002]. These systems take the importance of features into account and build a preference model for the user by utilizing any or a combination of methods like weighting the importance of features against each other (e.g., price is more important than color), weighting the preference on a value of a feature (e.g., red is preferred most, blue should be avoided) or what kind of items have been preferred in the past [Chen and Pu, 2012].

Demographic

A demographic recommender system provides recommendations based on the user's demographic data. Recommendations can be produced for different demographic niches by combining the ratings of users within demographic clusters [Ricci et al., 2011].

Hybrid Approaches

Some systems combine several above mentioned techniques to recommend items. Such systems are called *hybrid recommender systems*. They use advantages of one technique to mitigate the disadvantages of another [Ricci et al., 2011].

2.1.5 Preference Elicitation Strategies

In a recommender system, main subjects with diverse goals and characteristics are *users*, the objects that are recommended are *items* and important inputs for the recommendations are *transactions*. A transaction is the recorded interaction between the recommender system and a user that can be used by the recommender system algorithm to predict the user's preferences to generate better recommendations. Ratings, for instance, are the most popular type of transaction data that a recommender system collects and uses. A recommender system can either explicitly or implicitly elicit the user's preferences [Ricci et al., 2011].

Explicitly

Explicit preference elicitation can take on different forms. We can distinguish between unary ratings (e.g., "liking" an item on Facebook or the purchase of an item), binary ratings (e.g., item was rated good or bad), numerical ratings (e.g., providing 1-5 stars) and ordinal ratings (e.g., ratings on a 5-point Likert scale from "strongly disagree" to "strongly agree" with the middle being "neutral"). Writing product comments or tagging items (e.g., giving a clothing item the tag "casual") are other forms of explicit user feedback [Ricci et al., 2011].

Implicitly

Implicit ratings are collected by observing the user's actions to infer the user's interests. Examples for implicit preference elicitation is tracking the user's clicking behavior or time spent to consume an item, e.g., on a website [Ricci et al., 2011].

2.1.6 Iteration Steps of Recommender Systems

Recommender systems can also be distinguished by their number of iteration steps until the user has found an appropriate product. Developers of those systems should therefore consider the advantages and disadvantages of both interaction techniques.

Single-Shot Recommendation

Typical recommender systems, e.g., as known from a commercial system like the *Amazon* product recommendations, issue a one-time list of recommended items based on some initial preferences, maybe a search query. If the presented recommendations do not match the taste of the user, she has to issue another query to get a new set of recommendations [Smyth, 2007].

Conversational Recommender Systems

A recommender system can otherwise use a conversational approach [Smyth, 2007]. We can distinguish between three steps for interaction with conversational recommender systems [Woerndl and Lamche, 2015]:

1. Preference elicitation: Users can either explicitly state their preferences, or the system implicitly observes their behavior to create user models. For example collaborative filtering recommender systems often use ratings of users for items as input.
2. Result delivery and presentation: The computed set of recommendations has to be delivered and presented to the user on the corresponding device.
3. Feedback, critiquing and refinement: The user can select an item or give feedback on the recommended items in order to allow a refinement of the results [Woerndl and Lamche, 2015].

Due to the fact that our developed systems will use a conversational approach in order to improve the user experience, the following section will give a deeper insight into the characteristics and functionality of *conversation-based recommender systems* and in this context give special attention to *critiquing recommender systems*.

2.2 Conversation-Based Recommender Systems

A *conversational recommender system* supports an interactive process where the system, as well as the user, may provide or query information to each other [Ricci et al., 2011]. Three interaction steps can be defined: As already discussed before, the system first needs to elicit the user's preferences in order to generate personalized recommendations. Those recommendations then need to be presented in an ideal way to the user, appropriate for the used platform (e.g., for a mobile device). At last, there is a form of feedback required, to form an understanding of the user's preferences [McGinty and Reilly, 2011]. Different forms of feedback or rather strategies to navigate a user to her final recommendation can be employed in such a conversational dialog.

2.2.1 Navigation by Asking

The simplest form is by asking the user an array of questions. Learning from the answers, the system will ask follow-up questions to determine the user's interests. Good examples for navigation by asking are *ExpertClerk* by Shimazu or *Adaptive Place Advisor* by Goeker and Thompson which involve the user in a natural language, written, dialog [Shimazu, 2002], [Goeker and Thompson, 2000]. The author Smyth notes the importance of which and how many questions are asked to not endanger usability and push effort [Smyth, 2007]. Users in general dislike lengthy questionnaires, in particular if they do not understand what is asked or if they have to disclose sensitive or private information. Answering many questions in written form is also inappropriate on mobile devices and phones which often do not offer fast and easy text input methods in contrast to desktop systems [Smyth, 2007].

2.2.2 Navigation by Proposing

Another navigation approach is to show actual items and their properties right away and let the user express her interests. According to Smyth, this category includes three basic forms of feedback: *Ratings-, critiques-* and *preference-based feedback* [Smyth, 2007].

Ratings-based feedback allows the user to rate the items, e.g., "<x> out of <n> stars". This feedback method is commonly used in the previously discussed collaborative filtering systems.

Preference-based recommender systems assist the user within the search- and decision making process and can also be considered as a type of conversation-based recommender systems. When interacting with *preference-based feedback* systems, the user simply points out the one item she prefers over the others.

Critique-based (or *critiquing-based*) *recommender systems*, a type of conversational recommender systems, are more fine-grained and very popular nowadays. Here, the user can add constraints over features of a recommended item [Smyth, 2007]. Once a set of recommendations has been issued, recommender systems employing critique-based feedback allow the end user to make a change to a feature of a selected product. More clearly, the user is involved in a series of *critiquing interactions* until a satisfactory item is found. McGinty and Reilly argue that the primary reason why *critiquing* is a very popular feedback method in conversational recommender systems is based on the equilibrium of the effort a user has to put in and the quality of recommendations she gets in return [McGinty and Reilly, 2011]. Instead of having to specify exact values or sift through categories, users can simply state their current preference, e.g., "show me more trousers, but not in this color". Furthermore it does not require users to have a fundamental understanding of the item space when first using the system. Throughout the recommendation cycles and based on the effect of their *critiques* they become more familiar with those intricacies and can change and solidify their preferences based on the availability of certain product options. Most of all, critiquing enables the creation of very basic interfaces because it is a simple form of

feedback. There is no need for a category tree or a complex navigation interface [McGinty and Reilly, 2011]. Critiquing recommender systems can be classified into *system-suggested*, *user-initiated* and *hybrid critiquing approaches*, combining those two [Chen and Pu, 2012].

System-suggested critiques are, as the name implies, created by the system based on its knowledge of the item space, previously stated interests of the user and actual availability of items. The simplest form are a set of pre-defined, static critiques. The *FindMe* car navigator system illustrates the two available forms of static critiques (see *figure 2.1*). The user is able to, on the one hand, exactly specify the preferred value for each feature, e.g., price, motor power or number of seats. Those critiques on single features are called *unit critiques* [Burke et al., 1996, Burke et al., 1997].

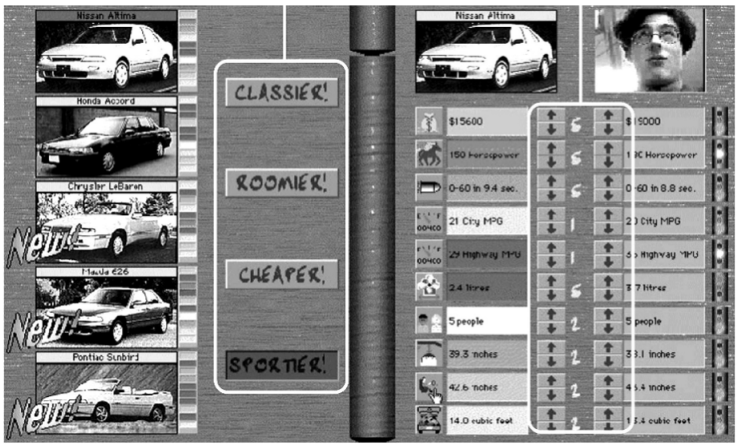


Figure 2.1: The *FindMe* car navigator interface. Highlighted are compound critiques on the left and unit critiques on the right [McGinty and Reilly, 2011]

On the other hand, there are critiques available that have an effect on multiple features at once, e.g., "sportier" would influence the power of the motor and the price. Those systems often just spell out the effect on features (e.g., "more motor power and higher price", see also *figure 2.2*). These are called *compound critiques* [Chen and Pu, 2012]. A disadvantage of system-suggested critiques is the imperfect accuracy on getting the user the product she actually wants. Chen and Pu note that available critiques might not always precisely match a user's interests, therefore causing the user to put in more effort to locate the item of her choice. In turn, this causes the system to be perceived as incompetent in helping to make quick and accurate decisions [Chen and Pu, 2012].

Instead of displaying a pre-defined set of critiques, younger research has focused on letting and encouraging the user to build her own critiques on any number of features in a trade-off like fashion, so-called *user-initiated critiques* [Chen and Pu, 2012]. This way a user can apply unit critiques or build a compound critique. The most notable example is the *Example Critiquing* system by [Chen and Pu, 2006]. It allows the user to choose one reference product in the beginning, then she can specify feature by feature whether to "keep", "improve" or "take any suggestion" on their value. Advantages of this approach are a higher level of user control as users can build their own critiques instead of being

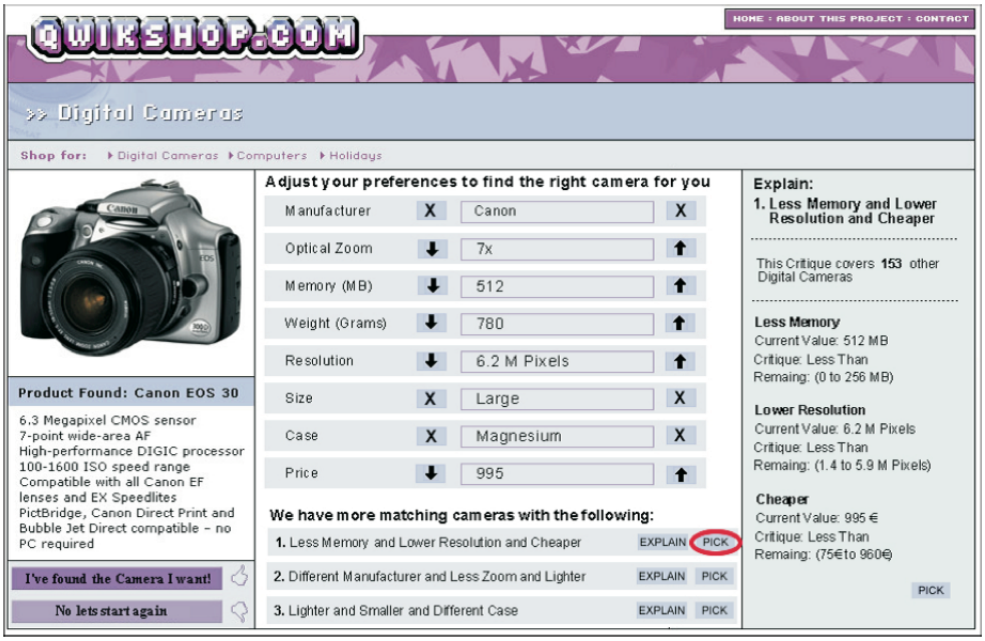


Figure 2.2: An online shopping interface using unit critiques (top) as well as system-suggested compound critiques (bottom) [Smyth, 2007]

forced to choose from system-generated options. It has been shown to have a positive influence on the decision accuracy and quality compared to a recommender using system-suggested critiques. The main disadvantage is the rather complex interface, users have to get familiar with first. The authors Chen and Pu created a table overview classifying some more systems based on the system-suggested and user-initiated critiquing variants [Chen and Pu, 2012].

Motivated by the issues of system-suggested and user-initiated critiques on their own, Chen and Pu created a *hybrid critiquing-based* recommender system [Chen and Pu, 2007a]. The first version combined the strengths from the user-initiated and the system-suggested critiquing approach. A follow-up version employed a preference-based grouping of recommendations where products are categorized by similar trade-offs regarding "improved" or "compromised" features [Chen and Pu, 2007b] . For instance, when recommending notebooks, one of these category titles could be: "*These products have larger display size, although they have shorter battery life*". The category title then served as the proposed compound critique with the option to show more items matching it. Both systems performed very well regarding subjective decision confidence, effort and trust in the system's abilities compared to a system using solely system-suggested critiques. This variant further improved effort in decision making and significantly reduced interaction time.

2.2.3 Combined Approaches

Obviously it is possible to combine the navigation by asking and navigation by proposing approaches to build a better system. Previously mentioned *ExpertClerk* switches from questions to proposing actual items after it has gathered sufficient information to do so, instead of just showing one final recommendation [Shimazu, 2002].

2.3 Evaluation Techniques

Traditional recommender systems have been concentrating on improving the accuracy. However, the focus nowadays lies on a broader set of measures and a more human-centered evaluation. Measuring user experience is challenging in general, and also in the case of recommender systems. Developers of recommender systems with a background in human-centered computing therefore need to think more broadly about both, the evaluation and the design of recommender systems and interfaces [Konstan and Riedl, 2012].

2.3.1 Overview

The authors Shani and Gunawardana distinguish between three different kinds of experiments: Offline, online and user studies. In offline experiments, an already existing dataset is used to estimate the accuracy of algorithms. Online experiments measure the system's performance on real users that perform real tasks. For example multiple algorithms can be compared by measuring the change in user behavior when interacting with these systems. In user studies, participants interact with the system by performing different tasks and are asked qualitative questions about their experience. Furthermore quantitative data can be collected such as the time needed to perform the task, the number of interaction steps and accuracy of the task results [Shani and Gunawardana, 2011]. Since the term "User Experience" describes an experience in the head of the user, the only reasonable solution is asking the user about her experience in order to acquire data about user satisfaction, perceived recommendation accuracy or effort and other feelings that are difficult to measure. Therefore, user studies are often employed for human-centered evaluations of recommender systems. In the past, a system's usability was tested after the product development has already been completed. Any changes to the user interface were cumbersome and its quality was therefore often treated with low priority. A so-called *user-centered design* process is gaining more attention nowadays. For example *SAP* established its usability lab in the mid-1990s and usability is now one of its main success factors [Butz and Krüger, 2014]. Corresponding evaluation methods can be distinguished between several dimensions. Our user studies are based on *empirical methods* because they focus on the interaction process between a user and a system, wheres *analytical methods* analyze the functionality and characteristics of a system. The quality of observed data can be described in terms of the following quality measures: [Gerrig and Zimbardo, 2008]:

Objectivity: The observed data is independent from the measurement methods or the observer.

Reliability: Results are not biased and repeated measurements by other researchers achieve the same results.

Validity: The observations measure exactly the variable that was intended and are representative for the general public.

It can be distinguished between two different empirical approaches: *Observational* and *experimental* methods. Observational studies just observe a naturally occurring phenomena without intervening or optimizing it. Test subjects are *naturally* assigned to different values of an independent variable. Within controlled experiments, the experimenter manipulates one or more independent variables and measures its effects [Field and Hole, 2002]. As we intend to evaluate different prototypes under certain aspects, we will conduct controlled experiments in our user studies. Statistical evaluation will be based on a one-tail paired t-test, ANOVA or a two-tailed paired Wilcoxon signed rank test. We will use a within-subject design to keep the number of testers at a reasonable size and a significance level of 0.05.

2.3.2 Metrics

Different metrics exist to evaluate the user experience. Standardized questionnaires can be used to measure the user experience of a product, such as *PANAS* (Positive Affect Negative Scale), *AttrakDiff* or *ResQue* tests. Also semi-structured interviews, in which the interviewer tries to elicit the psychological need that was the origin of the positive or negative feeling when using the product, might be applicable [Butz and Krüger, 2014], [Pu et al., 2011a]. The testing framework used for our user studies in the following chapters is a sub-set of the aspects relevant for evaluating critiquing recommender systems presented in [Chen and Pu, 2009]. Within this thesis, we will distinguish between *quantitative metrics* and *qualitative metrics*.

Quantitative Metrics

Quantitative evaluations deliver results that can be expressed in numbers, such as task performance times or error rates [Bortz and Doering, 2006]. In our prototype evaluations we will measure, among other, the following quantitative metrics:

Objective accuracy: This metric can be measured by showing each user at the end of each session and after answering all other questions a list of alternative items with similar attributes, sorted ascending by price. The user is then asked if she would go with any of those alternatives. The more capable the recommender is in guiding users towards her desired item, the lower this fraction will be.

Another way to measure objective accuracy is by calculating the *R-Score* which is based on the assumption that the value of a recommendation declines exponentially with the position of an item. The score for a user u, choosing an item i at position j is computed as follows [Breese et al., 1998]:

$$R_u = \sum_u \sum_j \frac{max(r_{uij} - d, 0)}{2^{(j-1)/(\alpha-1)}} \ . \tag{2.1}$$

r_{ui} refers to the rating of a user u for item i. Here, the rating r_{ui} is 1 if the user selected the item and 0 if not. A higher *R-score* refers to a better ranking of the item. d is a task-dependent neutral rating (here set to 0) and α is a half-life parameter which controls the exponential decline of the rating value (here set to 5, as recommended by Breese et al. [Breese et al., 1998]).

Number of critiquing cycles: As the system is a critique-based recommender system, the number of critiquing cycles is counted. More specifically, the counter is increased once the user has issued a critique, triggering the selection of new recommendations.

Time consumption: We also stop the time in seconds that passed from being shown the first set of recommendations towards selecting and confirming the final item.

Qualitative Metrics

Qualitative evaluations provide statements that cannot be expressed in numbers. Examples are answers of surveys, either via Likert scale statements or additional comments. Especially for measuring the user experience, qualitative metrics deliver interesting insights which could not be measured quantitatively [Bortz and Doering, 2006]. Within our user studies we will distinguish, among others, between the following qualitative metrics:

Perceived accuracy: To get an insight into how confident users are with their selected item, they were first asked if they were confident that the selected item was the best choice out of all items in the system. Clearly user were not shown all items, the important question here is if they felt to have seen all alternatives relevant to their specified preferences.

Perceived effort: Again, duration of time may not quantify if the user actually perceived the session as lengthy and tiresome. Therefore users are asked how easy they actually found the information (e.g., type of clothing) they were seeking and if it required too much effort.

Intention to return: From a commercial standpoint it is also important that users find a system useful enough to return to it at a later point. The best recommender system is useless if users do not see any value or benefit in using it. Participants are inquired about using such a system in the future if it were available (regular scale) and if they would rather never use such an app again (reverse scale).

These are our main metrics and most of them will be considered for the evaluation of our prototypes. However, several other qualitative metrics can be imagined such as *perceived trust, transparency* or *satisfaction* and will also be taken into account for some of our evaluations, depending on the prototype's goal.

2.4 Mobile Recommender Systems

With mobile phones becoming the primary platform for information access [Ricci, 2010], recommender systems also follow the trend of adaptation. However, a simple copy and paste from successful desktop-based recommendation systems cannot be straightforwardly performed for mobile devices. By information and interaction possibilities being available practically anywhere via wireless communication services (e.g., wireless LAN and GPRS/UMTS), so-called "ubiquity" mobile recommender systems offer more personalized and more focused content regarding the current user's context. Besides ubiquity, Ricci also mentions "location-awareness", e.g., the knowledge of users' current and past physical positions, as the second explicit property of mobile recommender systems contributing to more context-adequate recommendations [Ricci, 2010]. Location-awareness is one context factor of the type *physical context*. Several other context types exist, such as *social context* that represents the people that surround the user or *interaction media context* (also called *service-oriented context*) that describes the applications which are important for the user in the current context. Context-aware systems, meaning systems that have the ability to adapt to different contexts, can also be called *adaptive systems* [Butz and Krüger, 2014]. A mobile recommender system should therefore support and exploit this "user mobility", e.g., by storing a unique logical application so that the user's previous interactions will always be taken into account when calculating new recommendations, even when the user accesses the system with different devices or in another context. The mobile context should also be considered when generating recommendations, for instance suggesting a restaurant that matches the user's preferences as soon as the user checked-in into a hotel. Another property of mobile recommender systems is its "device portability". A user can access the recommender system with different devices and the user as well as the devices can move together. Connecting several devices to each other may produce more accurate recommendations, e.g., connecting a host running a web portal with a recommender system. However, new platforms bring new challenges. The challenge of showing only relevant recommendation information became even bigger because of the limited capabilities a mobile device has (compared to conventional systems built for desktop computers), such as the problem of a smaller screen, limited computing power, network availability, users behavior and impacts from external influences. Some of these restrictions have already been reduced to their smallest amount [Ricci, 2010]. Regarding limited screen size, an interaction technique specially designed for small devices, such as smartphones, is the so-called *peephole interaction* technique. This interaction technique allows the mobile user to view a large-scale content (e.g., a map) through a keyhole. This peephole is mostly *static*: Only the spatial layout behind the peephole can be moved (also referred as scrolling). More natural is a *dynamic* navigation: The peephole can be moved across the static spatial layout [Mehra

et al., 2006]. Several other technological developments in the smartphone sector compensate for the previously stated limitations and allow the development of new sophisticated mobile services. For instance, the latest generation of smartphones has not only powerful hardware, but also many devices come with built-in sensors that are used for precise environmental calculations, like location recognition via different position detection techniques (e.g., RFID or Wi-Fi beacon-based and GPS). The problem of network availability is with modern wireless broadband data transfer standards like LTE and low network access prices more-or-less (besides interruptions) minimized. Because of the development of these technologies and the incredible appeal of mobile devices and services there has been also much research and development work trying to apply recommendation technologies to this market. Another challenge when designing recommender systems for this mobile platform is the user's mobility. Since the user is on the move, connectivity problems might occur and the user's attention span is limited as well [Ricci, 2010]. Existing mobile recommender systems can be very diverse in terms of their application scenario, underlying architecture and recommendation methodology. A clear categorization of mobile recommender systems helps to differentiate the systems and to understand their specific characteristics. Previous works on mobile recommender systems use different classification approaches we want to discuss in the following subsections.

2.4.1 Classification Based on the Application Scenario

One possible way of classifying mobile recommender systems is according to the application scenario. Although researchers often try to generalize their work so that their systems can be adapted to several scenarios, most mobile recommender systems are designed for a specific application scenario. For example Krüger et al. present a review of mobile guide applications. They differentiate between three main application domains: *Museum guides*, *navigation systems* and *shopping assistants* [Krüger et al., 2007].

2.4.2 Classification Based on the Recommendation Occurrence

Moreover, Gavalas and Kenteris approach mobile recommender systems on the basis of the degree of user involvement in the occurrence of the recommendations [Gavalas and Kenteris, 2011]. Most recommender systems deliver recommendations in consequence of a user's request (*pull-based approach*). Nowadays, due to ubiquitous computers, recommender systems are capable to detect implicit requests and deliver recommendations on a push-basis (*proactive* approach) [Ricci et al., 2011].

2.4.3 Classification Based on the System's Architecture

The third classification criteria according to Gavalas and Kenteris is the system's architecture. The authors differentiate between *web-based*, *standalone* and *web-to mobile* recommender systems. *Web-based recommender systems* are usually based on a client-server

architecture. The server maintains the recommendation logic, whereas the mobile application client is part of the presentation tier. Full-fledged mobile applications are referred as *standalone systems*. *Web-to-mobile* recommender systems provide web-interfaces whereby users can personalize their application that can then be installed on a mobile device and used offline [Gavalas and Kenteris, 2011].

2.4.4 Classification Based on the Recommendation Algorithm

Another conceivable approach is the categorization based on the underlying recommendation algorithm, such as *collaborative filtering* and *content-based* recommendation (see *subsection 2.1.4*).

2.4.5 Classification Based on the Parameters Taken into Account

Gavalas and Kenteris present a classification based on the parameters taken into account to derive recommendations: *User constraints-based*, *pure location-aware*, *context-aware* and *critique-based* recommender systems. *User constraints-based* recommender systems use user constraints and preferences to generate the recommendations. The user's preferences can either be obtained *explicitly* through a short survey, or *implicitly* by observing the user's interactions. *Pure location-aware* recommender systems recommend items by considering the user's current location. Most systems of this category have been developed for early mobile devices, whereas GPS represented the solely context sensor. This classification is set in contrast to *context-aware* recommender systems. Within this category, the system's recommendation logic relies on a multi-dimensional contextual and situational space. These prototypes may in addition to the location consider more context parameters, such as the weather, the time and the user's current activity. *Critique-based* recommender systems solicit the user to criticize a recommended item through ratings to again issue an improved set of recommendations [Gavalas and Kenteris, 2011].

Since this thesis inter alia tries to investigate the question of which parameters, either derived from the environment or from the user herself, can improve mobile recommendations and how they can be collected in a user-friendly way, the last classification is a suitable approach to classify existing mobile recommender systems and at the same time focus on our research question.

User Constraints-Based Recommender Systems

Miller et al. implemented a distributed recommender system on a PDA. The system allows users of the *MovieLens* recommendation service to either select a video to rent or buy or find a theater nearby showing the preferred movie while away from the computer. The system now also offers offline access to the interface [Miller et al., 2003]. In a following version, the authors tackled the challenge to develop a recommender system that is both portable

and also protects the user's privacy. For that purpose they introduce an item-to-item collaborative filtering algorithm (*PocketLens*) that is based on a peer-to-peer architecture. In a first step, they built a model that captures the user-item relationship. This part can be done offline. In a second step, the model is used to compute a recommendation. The authors show that this peer-to-peer approach delivers fast and portable recommendations of good quality and still protects the user's privacy [Miller et al., 2004].

Pure Location-Aware Recommender Systems

The user's location and viewing duration of items can be used to generate recommendations of exhibits in a museum. Based on this idea, Bohnert et al. develop two different collaborative models to predict a visitor's interest in a museum: *Interest and transition*. The *Interest model* is based in temporal information. It considers the time the visitor spent at the exhibits to generate recommendations of unseen exhibits. The *Transition Model* implicitly collects spatial information. It calculates predictions based on the pathways followed by other visitors to the museum. In contrast to the unpersonalized *Transition Model*, the *Interest Model* adapts to the behavior of a visitor and is therefore personalized. As a third prediction model, the authors also consider a hybrid approach, a combination of the *Transition* and *Interest Model*. Results show that the *Transition Model* outperforms the *Interest Model*. The *Hybrid Model* produces the best performance. Nevertheless, it must be noticed that the experiments were conducted on a museum about marine life with homogeneous, arranged exhibits and it can be assumed that all visitors are interested in the topic marine life. Moreover, the underlying dataset is rather small and the researchers only considered the accuracy of the collaborative models instead of the user experience [Bohnert et al., 2008].

The collection of explicit feedback, such as ratings of locations, can be impractical and bother the user. Froehlich et al. investigated the relationship between explicit and implicit feedback concerning the travel behavior. Explicit rating of users determined the user's preferences. Implicit feedback such as visit frequency and travel time was automatically detected. In a user study with 16 participants, the users received a mobile phone loaded with the developed software. This software mobility sensor uses GSM signals and prompts participants to fill out a survey whenever the mobile device is stationary for a period of 10 minutes. Within these surveys, the participants were asked to answer specific questions about the place and how much they like it. Results show that there exist positive correlations between place preference and the implicitly measured data. Especially the combination of visit frequency and travel time correlates strongly with the user's place ratings [Froehlich et al., 2006].

GeoWhiz is a mobile restaurant recommender system that takes the user's current location into account. The authors assume that "*people who live in the same neighborhood are likely to visit the same local places (no one likes to travel)*" [Horozov et al., 2006, p.2]. They built their own database of about 12000 restaurants in the vicinity of Chicago to proof their hypotheses that there is a higher probability of a correlation between people

who live nearby than people who live further apart. However, it has to be differentiated between the user's current location and user's home location. Default correlation can only be made with users resident at a specific location. The authors claim that this conclusion has implications for the scalability of location-based recommender systems [Horozov et al., 2006].

Smartmuseum is a mobile cultural heritage recommender system that implements the idea of *context-based rating*. The system uses ontology-based user profiles. A user profile consists of several profile entries. Each of these profile entries include two different RDF triples: A triple (t), describing the user's interest and a triple (ct), describing the context. A user profile can either be created manually by using a web-interface: Here, the user explicitly adds concepts she likes or dislikes, such as places or persons she is interested in. The system then automatically expands these concepts as RDF triples. *Smartmuseum* also supports a dynamic user profile construction: If the user rates an item (either with *like* or *dislike*), the triple is attached to a context in which the rating process was performed. For example, if a user likes a renaissance-style painting and is located in Italy, the following triple would be generated: t = <sm:painting, sm:stylePeriod, koko:renaissance>, ct = <rdf:Resource, sm:userLocation, place:Italy>, w = 1. Each triple is also weighted based on the context (w∈ [-1,1]) which is observed from the tagging behavior of the user. Right now, *Smartmuseum* only uses location as a context but the context model is aimed to be expandable [Ruotsalo et al., 2013].

Context-Aware Recommender Systems

Compass is a context-aware mobile tourist application that recommends points of interest (POIs) based on the user's interests and current context. As soon as the user has expressed her interests or places she is looking for, recommendations of nearby buildings, buddies or other objects are generated and shown on a map and in a list. The application takes user dependent (such as location, speed, the user's schedule, last time an object has been visited and the user's shopping list) and user independent context information (such as weather, time or traffic information services) into account. An unsupervised online survey among 57 people investigated the usefulness of context-aware recommendations. Results show that most people appreciate recommendations that take the user's current context into account. The described approach shows how context-awareness can be integrated in a recommender system without describing a specific recommendation methodology. However, *Compass* requires the user to explicitly create a profile or to specify a goal [Van Setten et al., 2004].

Magitti is a mobile leisure recommender system that predicts ongoing and future activity from context and user behavior, so that the user is not required to explicitly define her profile or preferences. Its recommendations include stores, restaurants, parks and movies and are based on three key features: *Context-awareness* (current time, weather, location and store hours), *Activity-awareness* (distinguishes between five activity modes: Eating, shopping, seeing, doing and reading) and *Serendipitous, Relaxing Experience* (the user does not have to enter preferences, or queries). The recommendation logic relies on a collab-

orative filtering algorithm, the user's preferences and location and future plans that are derived from an analysis of personal data, such as calendar appointments, viewed documents and messages. Moreover it supports diversity by reducing the scores of items the user has already seen [Bellotti et al., 2008].

Wang et al. propose a heuristic approach, called *SCMSR* (more precise: Heuristic approach to **s**ocial network-based and **c**ontext-aware **m**obile **s**ervices **r**ecommendation), to search nearest neighbours for different users of a mobile services recommendation system. The mobile recommendations are based on both contextual mobile user preferences (such as morning/night or home/office), and mobile social network relationships. In a final evaluation on real-world data, the *SCMSR* algorithm is compared with a contextual prefiltering algorithm, traditional user-based collaborative filtering, and a basic matrix SVD (here, the latter two algorithms are the context-unaware baselines). Results show that *SCMSR* improves accuracy [Wang et al., 2011].

A two-phase context-aware proactivity model was developed by Woerndl et al. Phase one determines whether the user should receive a recommendation. Considering a recommendation system that suggests car drivers gas stations nearby, the system should delay the recommendation until the car stops for a moment (e.g., due to waiting at a red light). Only if a specific threshold is exceeded, the second phase will be initiated. In the second phase, the system generates scores for each item in the candidate set. Any recommender algorithm can be used for this purpose. The result will be a ranked list of items and the k best items will be displayed, depending on a second threshold. Finally, the user can give some feedback, similar to typical critique-based recommender systems. The feedback influences the thresholds. Both phases utilize different context parameters in the dimensions: *User context, temporal context, geographic context and social context* [Woerndl et al., 2007]. Based on this work, Woerndl et al. develop a context model to determine when to generate a proactive recommendation for mobile users. The focus is on the user context, e.g., the current user activity, user status and device status. An online survey serves to determine the appropriateness of a recommendation. Results enclose weights for each context factor that can be used as context parameters in the first phase of the proactivity model [Woerndl et al., 2012].

Gavalas and Kenteris introduced a mobile tourist guide (*MTRS*) that on the one hand takes the user's context into account (such as location, time, weather and already visited places) and on the other hand applies the concept of *context-aware rating*. The system increases weights of ratings provided by users using the mobile tourist guide application compared to ratings by web users, being away from the POI. MTRS uses a collaborative filtering based recommender system logic. However, new users are encouraged to register personal information (such as gender, age, tourist habits and interests) at the beginning so that they can be matched to one of the available generalized user classes (stereotypes) and enable personalized recommendations from right away. Moreover, *MTRS* proposes the use of wireless sensor network installations around tourist sites. This solution allows tourists to conveniently upload ratings or tourist information about POIs via their mobile device, without having to pay any roaming charges [Gavalas and Kenteris, 2011].

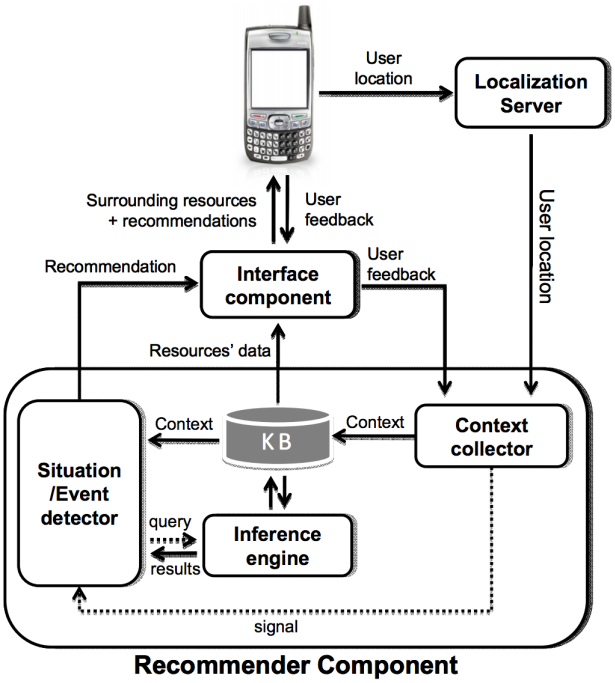


Figure 2.3: The system's architecture [Bouzeghoub et al., 2009]

Bouzeghoub et al. present a proactive recommender system to assist mobile users in a campus environment. The proactive system provides information about relevant buildings, individuals, events and available resources. Three components constitute the context: The current *user location* in the campus environment, the *user profile*, which includes the user preferences concerning their availability, their competences and interests and the *user agenda*, which stores the user's activities, including meetings, so that the system can automatically identify the user's activities and availabilities. The recommendation method is rule-based and consists of a *context collector*, *inference engine* and *situation/event detector* (see *figure 2.3*). The context collector uses ontologies to allow sharing and reusing the contextual information. It collects static (such as user profiles, agenda) and dynamic (such as interaction with the system and location) contextual information and stores this data in a *Knowledge Base*. The Inference Engine exploits semantic structures and allows inferring about concepts, attribute values, objects and their relationships. The situation/event detector uses the inference engine to analyze the current situation whenever the context collector signals that a user context has changed. A situation describes the context in an interval of time. For example, this expression states that Tom is following a conference this afternoon in amphitheater A5:

TomSituation = {in ($O_{User.}$Tom, $O_{Location.}$Amphitheater_A5), in ($O_{Location.}$Amphitheater_A5, $O_{Location.}$IST-campus), do($O_{User.}$Tom, $O_{Activity.}$conference), $O_{Time.}$thisAfternoon }.

If a situation changes, an *event* occurs. As soon as the recommender system detects such a change, the inference engine analyzes the rules in the knowledge-base. A recommendation is triggered if a match is found [Bouzeghoub et al., 2009].

Park et al. present a map-based mobile restaurant recommender system. The system is based on a Bayesian network that takes the user's personal information and context such as location, weather (e.g., cold and rainy) and time (composed of season and period) into account. The Bayesian Network is built by an expert while the parameters are learned by using training data. Each restaurant consists of thee attributes: *Class* (e.g., Italian or Chinese), *price* (e.g., mid-high) and *mood* (e.g., romantic or exotic). First-time users are asked to create a user profile with information such as name, age, gender, birthday, possession of car, blood type, monthly income, and food preference. The system automatically collects the contextual data. As soon as the user asks for a restaurant recommendation, a recommendation score is calculated. The preference regarding a restaurant consists of a weighted sum of the conditional probabilities of the class, price and mood attributes. The conditional probabilities are derived from the learned Bayesian network using an Expectation Maximization algorithm [Park et al., 2007]. However, the authors do not present a recommendation algorithm which automatically considers the weather and season attributes, but recommends restaurants for a specific time (breakfast, lunch or dinner) based on the previously indicated user preferences. Moreover, recommendation accuracy and user experience has not been evaluated.

I'm feeling LoCo is an ubiquitous mobile recommender system that recommends places nearby the user's current location, such as restaurants, museums or nightclubs by learning the user's preferences automatically and taking the user's mood into account. Physical context such as the user's current transportation mode and location are automatically detected. This physical information is used for a first filtering step: The user's mode of transportation and location influences the radius within which places for recommendations are considered. Faster movement implies a larger radius. The user's cognitive context is inferred based on the user's *foursquare* check-in history (a social network app to save and share visited places with friends[1]). Moreover, the user's mood influences the recommendations: *Foursquare* assigns each place to a category, which is mapped by the authors to a particular feeling: Arts & Entertainment (*feeling artsy*), College & Education (*feeling nerdy*), Food (*feeling hungry*), Home/Work/Other (*feeling workaholic*), Nightlife (*feeling like a party animal*), Great Outdoors (*feeling outdoorsy*), Shops (*feeling shopaholic*). As soon as the user selects one of these moods, the second filtering step is performed and only places assigned with the category to which the feeling is mapped to are recommended. The recommendation algorithm is based on text classification. The system considers the tags and categories associated with a place the user has visited. The user model is therefore a document, which holds all the names, categories and tags associated with a visited place. The four places with the highest log frequency weighting (a score that indicates how often a specific term is included in the tags of the user profile, as well as in the tags of the particular place) are recommended to the user. If not sufficient user data has been provided, *I'm feeling LoCo* mines the wikitravel page (wikitravel.org) of the user's current city for iconic

[1] https://foursquare.com

places. The system suggests this place to the user, if it is nearby and has the requested category on *foursquare*. A conducted user study shows that *I'm feeling LoCo* enhances the user experience and that the recommended places were overall satisfying [Savage et al., 2012]. This mood-based approach is in particular reasonable if a recommender system is aimed to suggest different types of leisure activities.

Discussion of Approaches and Research Gap

The recommender systems presented above have all one common goal: Delivering accurate mobile recommendations. However the user experience was mostly neglected when developing these systems. Although accuracy is essential for the success of a recommender system, the user experience is another important characteristic that gains more and more attention nowadays. Even a recommender system that generates very accurate recommendations may not be accepted by its users, because they have trouble or do not enjoy using it [Konstan and Riedl, 2012]. Previous authors came up with recommendation algorithms that improve the accuracy of mobile recommendations (e.g., by taking the mobile context into account such as in [Gavalas and Kenteris, 2011]) but what is missing so far are human-centered investigations that identify those characteristics that create a positive user experience. Most of the systems presented above have been developed for outdated mobile devices, so that specific properties of modern smartphones, such as touch-based interactions or new sensor technologies, which have the potential to improve the user experience, have not been investigated (e.g., [Park et al., 2007], [Miller et al., 2004], [Woerndl et al., 2007], [Van Setten et al., 2004], [Froehlich et al., 2006]). Some of the discussed approaches do not describe the interaction process at all and do not conduct a human-centered evaluation (e.g., [Horozov et al., 2006], [Bohnert et al., 2008], [Wang et al., 2011], [Woerndl et al., 2012]). The work of Bouzeghoub et al. made some considerations concerning the human-system interaction process, however they only present an abstract approach without evaluating it [Bouzeghoub et al., 2009]. Savage et al. considered the user experience when developing the system *I'm feeling LoCo* [Savage et al., 2012]. However, they did not specifically analyze which characteristics of a recommender system effect a positive user experience. Ruotsalo et al. and Bellotti et al. tested their mobile recommender system also within a user study to evaluate the user experience [Ruotsalo et al., 2013], [Bellotti et al., 2008]. However, they only took the user's location and explicitly stated preferences into account and we believe that even more aspects have the ability to create a positive user experience of mobile recommender systems. We also believe that a conceptual framework that includes all features that have the potential to improve the user experience of a mobile recommender system, will help developers to commercialize their applications. We want to help preventing that a great recommender system with a high accuracy is not accepted by its target group because of its negative user experience. Also Shneiderman and Plaisant point out that applications for mobile devices still need to be custom designed and that a framework or guidelines are necessary in order to develop appropriate user interfaces for mobile devices and also take advantage of its specific properties [Shneiderman and Plaisant, 2005]. This conceptual framework should on the one hand consider all additional challenges mobile recommender systems are facing compared to desktop systems,

on the other hand, HCI aspects need to be studied to understand how user experience is actually created in a mobile recommender system. By analyzing the different steps of the interaction process between users and mobile recommender systems, we will identify which steps actually influence the user experience and how they can be optimized. We will come up with solutions for each step of the whole recommendation process tailored to a mobile device either based on own considerations or based on suitable solutions of related works that we will study more detailed in the next chapters. Developers of mobile recommender systems that take our proposed solutions into account can then concentrate on the accuracy of their systems without being concerned about the generated user experience.

2.5 Conclusion and Next Steps

This chapter introduced the fundamentals of (mobile) recommender systems. Starting with giving a definition of the term "Recommender System", we discussed its functionality and history. One main focus was on conversational recommender systems because we believe that this approach is suitable for a mobile recommender system and generates a positive user experience. We explained their characteristics and presented different variants. The next section discussed several evaluation techniques and presented the testing framework for our future evaluation of a mobile recommender system's user experience. We concluded the chapter by discussing related work and verifying our research question. Within this chapter we only targeted a scientific perspective on mobile recommender systems and identified a research gap. The subject of the next chapter is therefore to propose a solution to this problem by developing a conceptual framework which helps to improve the user experience of mobile recommender systems.

3

Framework for the Generation of Mobile Recommendations

The goal of this chapter is to investigate which human-computer interaction (HCI) aspects have to be considered when developing a user-friendly mobile recommender system. The user experience of a system plays a crucial role for the success of a recommender system. Based on previous research we therefore determine the most important aspects mobile recommender system developers should take into account to enable a positive user experience. We present a conceptual framework which illustrates these aspects and their interactions. The first section (1) introduces the motivation behind this framework. The second section (2) presents previous research focusing on HCI aspects of recommender systems. Those results define the conceptual framework which is explained in the third section (3). The next section (4) discusses possible application scenarios and justifies the selection of the main scenario in this thesis. We finally give a conclusion and an outlook on the next research steps in the final section of this chapter (5).

3.1 Motivation

The user experience and usability of a system play a crucial role for the success of a recommender system [Konstan and Riedl, 2012]. Although the analysis of related work in the previous chapter showed that some research was already conducted in the scope of mobile recommender systems, the main goal was to improve the system's accuracy instead of the user experience. However, mobile systems reveal additional challenges related to human-computer interaction issues that have not been researched thoroughly: First, supporting input and interaction capabilities on mobile devices to elicit user preferences is extremely difficult because of their spatial limitations in the user interface (e.g., small keypads and screens). Second, users might be unable to formulate explicit queries and prefer being involved in an *exploratory* process due to their uncertainty at the beginning

of the recommendation session (see *section 2.1.1*). Third, since the user is on the move, connectivity problems might occur and the user's attention span is limited as well [Ricci, 2010]. As discussed in the previous chapter, the evaluation of the user experience is a user-centered design process. To tackle those challenges, we therefore analyze related work in the HCI area and come up with a conceptual framework which helps to improve the user experience and at the same time the accuracy of a mobile recommender system. HCI research nowadays focuses not only on how humans perform tasks but on the collaboration, connection, emotion and communication of the user with the system. Whether people like using an interface (meaning a positive user experience is generated) is getting more important than the system's efficiency. HCI is therefore a research area that has to dynamically adapt to new research methods and technologies. In conjunction with Ben Shneiderman's statement "*the old computing is about what computers can do, the new computing is about what people can do*" [Shneiderman, 2015, p.1], Lazar et al. point out that HCI combines several fields that involve the study of people, such as how they think, learn, communicate and how objects should be designed to meet their needs [Lazar et al., 2010]. The aim of the framework is to support developers of mobile recommender systems by giving concrete solutions and guidelines how to consider and implement those essential HCI aspects that have the potential to improve the user experience of a mobile recommender system so that it can succeed in the market. The framework consists of different building blocks which may influence each other in some way. Due to the fact that most existing user studies that took the user experience of a recommender system into account, have not considered a mobile scenario, we have to investigate if the building blocks can be applied to a mobile recommender system and how this can be done. Moreover, we have to study if the interaction of all these building blocks still creates a good user experience when combined in one system.

3.2 HCI Aspects of Recommendations in Previous Research

Anand and Mobasher define a framework that integrates context to generate accurate recommendations from a system logic perspective (see *figure 3.1*). They distinguish between a user's *short term* (STM) and *long term* memory (LTM). STM stores explicit or implicit ratings for items from the active interaction. For example the users' ratings of items, the time spent on each page, the features of items viewed, semantic properties of items of interest, search queries, or other implicit or explicit measures of interest. LTM stores preference models within specific contexts, derived from previous user interactions with the system. Contextual cues are used to retrieve relevant preference models from LTM that belong to the same context as the current interaction. Cues extracted depend on various factors such as the users' ratings of items (*collaborative cues*), the semantic properties of objects of interest as might be available through a domain ontology (*semantic cues*) or the amount of time spent on a page (*behavioral cues*). This information is merged with the current preference model stored in STM for generating context-aware recommendations to predict ratings for items not currently rated by the user [Anand and Mobasher, 2007].

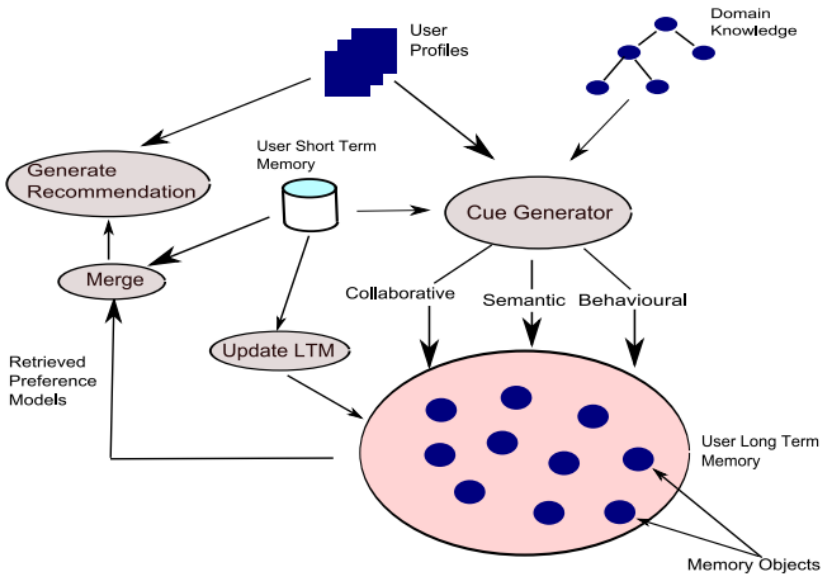


Figure 3.1: A context-aware recommendation process [Anand and Mobasher, 2007]

Although context plays an important role regarding the perception of the usefulness of an item for a user [Ricci, 2013], the proposed framework is very general and does not emphasize how it can be applied in a mobile scenario. First of all, mobile recommender systems face a different context than web-based systems that has to be treated differently. Ricci divides mobile context into four sub-categories: *Physical context* (e.g., location, time, weather), *interaction media context* (e.g., the device used to access the system, the type of media that is being interacted with), *modal context* (e.g., the user's intention, mood, experience) and *social context* (e.g., people that surround the user) [Ricci, 2013]. A concept for the generation of mobile recommendations should include these four different types of context.

⇒ **Building block 1:** *Mobile Context*

Due to spatial limitations in mobile interfaces, the efficient presentation of mobile recommendations is difficult. Interactive interfaces have to be designed especially for mobile devices to support the elicitation of the user's preferences. This should be done in an intuitive and user-friendly way, so that users understand the system's functionality right from the start and enjoy interacting with it. Ensuring that the level of cognitive and interaction effort is kept to a minimum supports a smooth user experience [McGinty and Reilly, 2011]. Pu and Zhang also emphasize that the presentation of the items has a high influence on the user's perception and should be designed wisely [Pu et al., 2012].

⇒ **Building block 2:** *Presentation*

Explanations of recommendations represent another HCI aspect that should be taken into account when developing mobile recommender systems. Recent user studies found

out that users of recommender systems appreciate transparent and easily understandable recommendations [Tintarev and Masthoff, 2012]. Automatically generating explanations in a mobile recommender system is also emphasized by other researchers in order to support a positive user experience [Ricci, 2010], [Konstan and Riedl, 2012]. However, this topic has received very few studies so far.

⇒ **Building block 3:** *Explanations*

For the purpose of delivering accurate recommendations, the recommender system needs to learn about the user. This is a difficult task since the user is often uncertain of her preferences, in particular in an exploratory scenario where the user does not know exactly what she is searching for (e.g., looking for an open restaurant nearby). Moreover, due to a limited attention span and spatial limitations in mobile interfaces, the user might be unable to formulate explicit search queries. *Conversational-* or rather *critiquing recommender systems* aim at solving this problem (see *section 2.2* for more information). Once a set of recommendations has been issued, it will involve the user in an ongoing conversation and solicit a user critique through ratings to again issue an improved set. Instead of having to specify exact values or sift through categories, users can simply state their current preference, e.g., "show me more trousers, but not in this color". Furthermore it does not require users to have a fundamental understanding of the item space when first using the system [McGinty and Reilly, 2011]. Konstan and Riedl also emphasize that putting the user in control over how the system functions is one of the key criteria of recommender systems that create a positive user experience and therefore suggest an interactive approach [Konstan and Riedl, 2012]. It is important that the recommender system presents on the one hand items that are interesting, on the other hand items that will result in enhanced knowledge about the user, as soon as the user provoked a critique. Especially for the elicitation of the user's preferences for mobile devices Rubens et al. suggest the concept of conversation-based *Active Learning* but without investigating the mobile application area further. By starting with general recommendations, the system narrows down the user's interests through eliciting critiques until the desired item is obtained. In contrast to other existing systems, even new users receive recommendations right from the beginning. The user can always interrupt this process when distracted and only simple interaction methods are needed to exercise a critique [Rubens et al., 2011].

⇒ **Building block 4:** *Active Learning*

The underlying user model plays an important role for the user-friendliness of a recommender system. The framework designed by Anand and Mobahser (see *figure 3.1*) also considers this aspect, however they do not specify how a user model can be quickly built to deliver recommendations right from the start. In order to provide personalized mobile recommendations even in the cold start phase, a user modeling approach based on *stereotypes* seems to be a promising approach [Rich, 1979b]. Depending on the application scenario, most people can be associated with a specific stereotype. For example in the fashion domain some people have a sporty, some have a rather elegant fashion style, so that stereotypes can be easily predefined and an already existing user data base is not required. Stereotypes may also apply on other mobile scenarios. Users of a restaurant rec-

ommender system may for example either favor a health-conscious, an exotic or a hearty cuisine. Within a stereotype-based user model, new users will be initially assigned to different categories to receive personalized recommendations right from the beginning.

⇒ **Building block 5:** *User Modeling*

Based on the previous considerations, these five building blocks are essential for designing a mobile recommender system with a positive user experience from an HCI perspective: *Mobile Context, Presentation, Explanations, Active Learning* and *User Modeling.*

3.3 Conceptual Framework and Expected Contributions

The literature review has shown that very little research has investigated the development of a concept which describes the overall recommendation process of mobile recommender systems with a positive user experience. In order to come up with such a concept, we take the framework of Anand and Mobasher as a basis. However, this framework only considers the main building blocks *Context* and *User Model* (in *figure 3.1* divided into sub-building blocks) and ignores the interaction process of a recommender system that puts the user in control (e.g., a conversation-based system). As research has shown, this is an essential characteristic of a recommender system that creates a positive user experience [Konstan and Riedl, 2012]. When taking a look at the interaction processes illustrated by several researchers in the area of conversational recommender systems (e.g., [Viappiani et al., 2006] and [Pu et al., 2011b]) they all use a basic concept: First, the user's initial preferences are collected. Then, the system generates and presents recommendations to the user. As long as the user is not satisfied with an item, the system elicits the user's feedback (in our case "critiques") and updates the recommendations. *Figure 3.2* shows the basic user-recommender system interaction process.

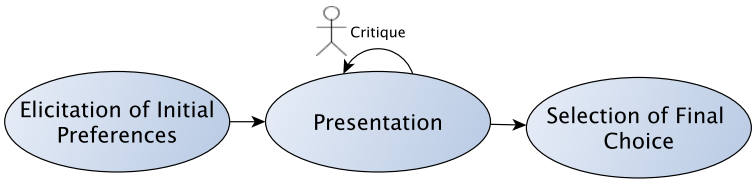


Figure 3.2: A basic user-recommender system interaction process

We now combine this interactive process with the context-aware approach presented by Anand and Mobasher and, as a result, gain a conceptual framework that considers all characteristics that research has identified as being the major source of positive experience with mobile recommender systems (see *figure 3.3*). The framework and its expected contributions have also been published in [Lamche, 2014]. Just like in the work of [Anand and Mobasher, 2007], it adapts the idea to distinguish between a short term and a long term memory. However, we define the short term memory as an instance that stores the overall mobile *Context* (consisting of the modal, physical, social and media context). This *Con-*

text is used to update the long term memory (here defined as the *User Model*) after each interaction. The building block *User Model* includes the component "elicitation of initial user preferences" due to the stereotype-based approach, which elicits the user's initial preferences in the beginning. The *User Model* also includes the user profile, since this is data that is constant (such as the user's nationality, gender and date of birth). The information of the *User Model* is then used in conjunction with the *Context* as input for the *Active Learning* algorithm that generates the recommendations and makes the elicitation of the user's critiques possible. We combine the interaction of the user with the system in the "critique-arrow". It should be mentioned that the different feedback elicitation strategies will be discussed when investigating the *Presentation* building block. The recommendations will be presented in an understandable and transparent way, so that the user can criticize them efficiently, which in turn updates the modal context. For this purpose, suitable explanations have to be generated and presented. The *Explanation* building block can be seen as a part of the *Presentation* component, however we decided to investigate it individually since there exists almost no research regarding the generation of explanations for mobile recommender systems. Also other authors separate the *Explanation-* from the *Presentation* component within a user-recommender system interaction processes (e.g., [Ricci et al., 2005], [Jannach and Kreutler, 2007]).

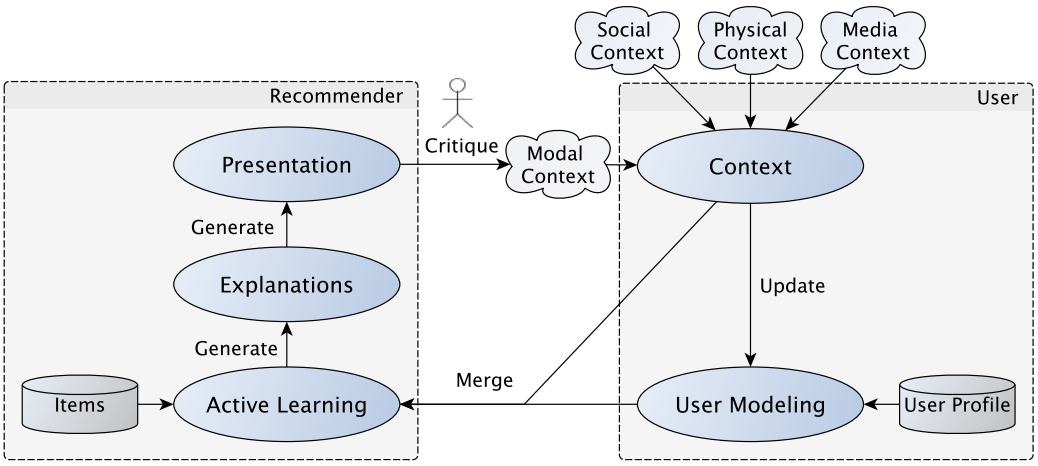


Figure 3.3: A conceptual framework for the generation of mobile recommendations

The goal of the framework is to provide findings about how to improve the user experience of mobile recommender systems by the application of human-computer interaction methods. Expected contributions will be fivefold:

Contribution 1: Investigation of different Active Learning strategies and presentation of a customized algorithm specifically for exploratory mobile scenarios.

Contribution 2: Proposal of a suitable user model for mobile recommender systems based on stereotypes.

Contribution 3: Guidelines for the design of interactive mobile recommender user interfaces.

Contribution 4: Better understanding of mobile context and how it can be used to improve mobile recommendations.

Contribution 5: Detailed understanding of how explanations should look like and how they can be generated automatically.

The final evaluated conceptual framework should give mobile recommender system developers an understanding of how to generate user-friendly and accurate recommendations in mobile scenarios. With this knowledge, mobile recommender systems can be developed more quickly and successfully. We want to point out that although we consider this framework as generalizable for mobile recommender systems with a positive user experience, we will restrict the investigation of the framework in the following ways: Regarding the consideration of feedback, we will mainly focus on explicit user feedback and neglect implicit feedback. This is due to the fact that putting the user in control and thereby increasing the system's transparency has higher chances to generate a positive user experience than when the system uses more noisy data by collecting feedback implicitly [Konstan and Riedl, 2012]. Moreover, we will base our user model on stereotypes. Different approaches to build a user model can be imagined, however, we consider a stereotype-based approach as suitable for a mobile scenario as it can be build quickly and in a user-friendly way.

3.4 Possible Application Scenarios

Various mobile scenarios can be imagined as application domains. For tourists, a recommender system suggesting points of interests or restaurants might be a possible application scenario. Looking at a persuasive recommender system that promotes healthy lifestyles and improves health information, a mobile system that presents dietary recommendations concerning restrictions on, e.g., consumption of sugar (for diabetics), fat (for overweight people) or for people with specific allergies or intolerances might be another scenario that benefits of such a framework. Due to the fact that it is a time-consuming process to build an appropriate dataset for the particular user studies, we have to decide for one common application scenario. We choose a mobile shopping recommender for the following user studies. The reason is that in particular for exploratory scenarios such as going shopping without having a specific item in mind, a system that adapts to the consumer's preferences constitutes a domain that is largely unexplored in the literature. Moreover, clothing items can be tied to shops currently open near a user's position and allow simple critiquing of various aspects like color and price. As the system allows multiple types of clothing, the data is also very diverse in nature.

3.5 Conclusion and Next Steps

Within this chapter a conceptual framework for the generation of accurate and user-friendly mobile recommendations was presented. This framework aims at describing all necessary steps that need to be implemented when designing an efficient mobile recommender system from a human-computer interaction perspective. As shown in *figure 3.3*, the framework consists of five building blocks that require further research: *Active Learning, Explanations, Presentation, Context* and *User Modeling.* All of these building blocks reveal open research questions and therefore will be evaluated separately, both theoretically and empirically. For each experiment, we will develop a prototype so that the theoretical considerations can be evaluated within a user study. The outcome of each of the experiments will iteratively extend parts of the framework for mobile recommendations. The individual five building blocks of the conceptual framework will be investigated separately within the next five chapters and combined in an overall evaluation. This evaluation provides information about the correct interaction and compatibility of the building blocks. As a final result, we will come up with an evaluated framework consisting of presentation guidelines, efficient algorithms and suitable user modeling approaches that support developers to improve the user experience and accuracy of mobile recommendations.

4

Active Learning

This chapter reflects the implementation and evaluation of a mobile shopping recommender system using Active Learning. It will give insight on the reasoning behind why this specific Active Learning algorithm was chosen, detail the development and design process and show how two variants of the system performed against each other in a real user test. The chapter is divided into four sections. The current one (section 1), introduces the ideas, motivations, existing research and goals of this chapter. The second section (2) explains our developed prototype – a mobile shopping recommender system that integrates a conversation-based Active Learning strategy. It is split into an algorithm part focusing on recommendation retrieval and feedback and a design part about the interface and interactions. We then present in the fourth section (3) the results of our user study that showed that our approach enhances the user experience and that users prefer diversity based Active Learning to similarity based Active Learning. It elaborates on the goals, methods and testing framework used as well as the dataset and test hardware. The final section (4) summarizes the achievements and hints towards further development and research potential.

4.1 Motivation

The focus of this chapter is on *Active Learning* recommender systems, which can quickly deliver personalized results without preexisting data about the user (which can be called *Passive Learning*). While a traditional recommender system suggests items based on existing data, e.g., previous or other users ratings that have accumulated over time through interacting with the system, recommender systems using Active Learning will actively decide which items (called training points) to show the user and based on her reactions get a better idea of her preferences [Rubens et al., 2011]. The concept of Active Learning has been applied to recommender systems in general, but was not particularly used as a means to tackle the shortcomings of mobile recommender systems in terms of interface limitation and user's preference elicitation. Especially for exploratory scenarios, where

the user does not know exactly what she is searching for (this may include for instance restaurants or clothing items), recommending items from the beginning without requiring the user to insert a search query is important (see also *2.1.1*). Therefore a context-aware information retrieval is important for each session. The goal of this chapter is to explore if Active Learning strategies that also consider the user's spatial context can improve the user experience of mobile recommender systems.

4.1.1 Conversation-Based Recommender Systems

We focus on *case-based* recommender systems where items are represented as a set of features (see also *subsection 2.1.4*). In order to elicit the user's preferences, we apply a *conversation-based* approach (see *section 2.2* for more details). A conversation-based recommender system that first recommends items to the user and then asks for feedback (*navigation by proposing*) can be divided into three basic steps. To begin with, recommendations or cases have to be retrieved from a larger dataset. Those recommendations then need to be presented in an ideal way to the user, appropriate for the used platform, in the case of this work a smartphone. At last, there is a form of feedback required to form an understanding of the user's preferences. We will use a critique-based feedback approach.

Conversation-Based Active Learning

Standard Active Learning methods focus on gathering ratings on training items to improve the prediction of a user's ratings of other, yet unrated, items. Conversation-based Active Learning starts from a general selection of items and involves the user in a series of interactions (recommendation cycles) that narrow down the item space to recommendations that closely match the user's interests. The item space is already constrained to a smaller set, e.g., using context data such as location or time when recommending a clothing store, from which the system will try to suggest items with the highest utility regarding the user's current preferences. This approach works great for users who do not know their preferences right in the beginning, but rather form them over time while using the system: It allows for exploration (e.g., a diversity-based Active Learning strategy) and helps the user become aware of her interests. Therefore, conversation-based Active Learning is a promising approach for a mobile recommender system and serves as a basis of our proposed system.

There are two distinct variants on how to enable a conversation with a recommender system. A case-based method, which suggests new recommendations based on feedback through e.g., critiquing on current recommendations. And a query-editing method, which uses a search query, that is refined, optionally with the help of some advisor, to get a better set of suggestions. To hide recommendations behind entering terms into a search field (or even a set of initial questions) conflicts with the vision of quick results and is less suitable for a mobile recommender system [Rubens et al., 2011].

Similarity vs. Diversity-Based Active Learning

Traditionally, conversational recommender systems suggested items that are maximally similar to the user query. Recent research has shown that this technique affects the user experience because it limits the coverage of the recommendation space. Clearly similarity, as well as diversity (considering dissimilar items) have both to be taken into account equally while building the set of recommendations. Diversity is also important in an exploratory scenario where the user is looking for an inspiration for a present or if the user is looking for a restaurant and prefers a varied cuisine. For this purpose, Smyth and McClave introduced a quality measure which is used in their *bounded greedy algorithm*. The algorithm incrementally builds a retrieval set, R. The amount of items so far selected is denoted by m ($R = \{r_1, ..., r_m\}$) while n refers to the amount of items in the case base C. During each step the remaining items are ranked according to their quality with the highest quality item added to R. The quality of an item c is proportional to the similarity between c and the current target item t, and to the diversity of c relative to those items so far selected ($R = \{r_1, ..., r_m\}$). To be more precise, the algorithm first selects the item with the highest similarity to the search query. During each subsequent iteration, the item with the highest combination of similarity and diversity is selected with respect to the set of items selected during the previous iteration. The authors also introduce a weighting factor α that allows biasing the quality metric in favor of either similarity to the target query or diversity among selected cases. This metric is used in their *Adaptive Selection* algorithm discussed in detail later. The relative diversity between cases is defined as the average dissimilarity between all pairs of cases. What follows are *equations 4.1* and *4.2*.

$$Quality(t, c, R) = \alpha * Similarity(t, c) + (1 - \alpha) * RelDiversity(c, R) \tag{4.1}$$

$$RelDiversity(c, R) = \begin{cases} 0 & if R = \{\} \\ \frac{\sum_{i=1}^{m}(1 - Similarity(c, r_i))}{m} & otherwise \end{cases} \tag{4.2}$$

To reduce computational burden Smyth and McClave only apply the quality metric in *equation 4.1* to the target query $b * k$ most similar cases ($b > 1$). k refers to the amount of items that are recommended to the user. This refinement has a greatly reduced retrieval cost since k items are selected from only $b * k$ cases instead of from all n items and $bk \ll n$ for in general low values of b and k. b denotes the *bound* and should be selected wisely. For a suitable bound b it can be ensured that it is unlikely that cases with a potentially higher quality metric are ignored, because they are not within the top $b * k$ most similar cases [Smyth and McClave, 2001].

4.1.2 Existing Approaches

Some research approaches already tried to elicit the user's preferences by actively involving the mobile user in a conversation or by collecting the user's context, so that good mobile

recommendations can be provided right from the beginning, without requiring the user to insert a precise search query.

Nguyen et al. were among the first researchers to tackle this problem. They point out that most critiquing-based recommender systems are built for desktop computers and are often web-based. Since mobile devices are becoming the primary computing platform for consumers, they developed a mobile recommender system that displays recommended restaurants in a simple list. Users request a map-based interface to get a better idea of their position relative to the provided suggestions and to help them to actually get there in the end. Location-awareness and in general awareness of context have come to be valued by users in those mobile situations. Moreover, they introduced colored icons to show how good a match of a recommendation is regarding the user's preferences, to help the user to quickly realize the appropriateness of a recommendation, what reduces interaction time. It also enables a quick comparison opportunity between items and lets the user more easily see the effect of her critiques after each recommendation cycle [Nguyen et al., 2004].

Based on this system, Ricci and Nguyen developed their conversation-based *MobyRek* application. Their main issues to adapt to the mobile form factor were the small screen size and the limited possibility for input that make entering of queries time consuming and complex. The algorithm ranks items according to their similarity to the elicited preferences. The user can choose between three different search initializations: "Use my profile", where the system constructs the search query by exploiting an already existing user profile, "let me specify" and "similar to" [Ricci and Nguyen, 2007]. However, new users always have to specify queries and since this approach is similarity-based, users should already have a concrete restaurant in mind. The restaurant descriptions are solely text-based and do not include images of the recommendations.

A very simple move to decrease the cognitive effort is to move to a more visual representation instead of relying solely on text. For example Zhang et al. represented compound critiques through a set of colored icons. Their research shows that users are more likely to engage with the more visual compound critiques and interaction time with the system was reduced. The icon approach also enables to glance this information quickly on smaller screens; appropriately their system was optimized to run on the newly released iPhone later on [Zhang et al., 2008]. In this approach, users always have to specify the preferred features before getting recommendations for technical devices such as cameras or laptops.

Braunhofer et al. present an Active Learning based approach for a points of interest mobile recommender system called *STS* (South Tyrol Suggests). The user is first asked to fill in a personality questionnaire which comprises questions such as "*I see myself as open to new experiences, complex*". The system then uses a matrix factorization model to predict items that the user might be familiar with and asks for her rating. By rating an item, the user also explicitly states the contextual situation, in which she experienced the item in the past. A live user study comparing the new personality-based approach (*Personality-Based Binary Prediction*) with an approach that takes the item's popularity into account (*Log(Popularity) * Entropy*), is conducted, involving 51 people. Results indicate that both systems acquired ratings that resulted in recommendations that fitted the users'

preferences, but only the new approach generated recommendations that were significantly rated as "better chosen" [Braunhofer et al., 2014]. However, this approach is not critique-based, the user can only influence the recommendations by completing the questionnaire and rating the items, but cannot state which features of an item she particularly likes or dislikes.

Viscors is a conversational mobile recommender system for wallpaper images. The system combines two information-filtering techniques: Collaborative and content-based filtering. Recommendation logic consists of several steps. First, a collaborative filtering algorithm produces the initial list of recommended wallpapers. If the customer is using the system for the first time, a best-seller-based method is applied, instead of collaborative filtering. Then, the customer can either select an image to purchase it or use it as a query for a content-based search of similar wallpapers. For the computation of similar images, the system calculates for all images in the database the distance from the query and recommends the most similar wallpapers, one by one. For each image, the user declares whether she likes it or not. These iterations are repeated until the customer purchases a wallpaper or quits the process. Results show that this approach can significantly decrease the views per success, compared to pure collaborative filtering or the best-seller-based algorithm [Kim et al., 2004]. However, this non-critiquing-based approach only allows the users to influence the recommendations by selecting a wallpaper that is approximately satisfying. A new set of items is therefore calculated based on similarity.

Having a closer look at the above mentioned recommender systems, it becomes obvious that none of these approaches combine the following three aspects, which are essential for an exploratory mobile scenario: First time users should have the possibility to *criticize* recommendations right from the beginning, without having to specify a preference or a search query. Second, the user should be involved in a *conversation* to *criticize* each product feature either positively or negatively (e.g., "I do not like red-colored items") in contrast to specifying her preferences in the beginning. Finally, the mobile recommender system should support *diversity-based recommendations*. We hypothesize that diverse recommendations support exploration and avoid over-specialization. Therefore, instead of the establishment of a permanent user profile, context-dependent information retrieval should be supported for each session.

4.1.3 Goals

Due to the final goal of creating a mobile shopping recommender system using Active Learning strategies on a smartphone, multiple properties arise that the final system should fulfill. As pointed out before, interactions with a mobile device should be fast and fluid to integrate well into typical day to day usage. Therefore the system should require little effort to use and minimize user frustration. It shall have an easily understandable interface with quickly graspable visuals like icons and images and only use text where necessary. It should not require the user to specify initial preferences through filling out questionnaires or typing search terms but content should be shown right when opening the application.

However, initial restrictions based on location and time may be imposed by the system itself without requiring any user interaction. The system should be based on interfaces and interactions designed for a modern smartphone's large (4+ inches) and high definition resolution (about 1280x720 pixels and up) display. It shall further leverage and optimize for touch-based interaction for easy and fast input. On each use, the system should adapt to the users current needs. It should not rely on a static profile or other users' ratings through collaborative filtering. It should enable exploration of the search space as well as exploitation of the user's preferences to deliver more relevant recommendations. The Active Learning algorithm, interaction- and interface design should be based on and combine existing research in this field. The evaluation of the system should ground on a real-world scenario. The system itself should resemble a real world product, a modern smartphone application, short of few modifications and conditions. The employed Active Learning algorithm should be available in two variants, one serving as a baseline for evaluation. These two variants should be tested with a diverse set of real world users and diverse real world data. Results should indicate general positive and negative points about the mobile recommender system as well as the advantages or disadvantages of one algorithm variant over the other. Evaluation criteria and processes should also be based on and combine existing research of critiquing recommender systems. To sum up, it is sought to answer the following questions with this thesis on the task to implement and evaluate a mobile shopping recommender system using Active Learning strategies:

Research Questions: How should the recommender system handle the limited screen size? Which critiquing algorithm should be applied? Is the implemented recommender system performing well and is it well received by real users?

Based on this, the thesis will also indirectly answer if Active Learning is a good approach for a mobile recommender system.

4.2 Designing the Test Application

The goal of the test application is to combine Active Learning methods with a mobile shopping recommender system developed for clothing items that also considers the user's spatial context, to test if users prefer recommender systems that show items similar to previously liked items (*SIM variant*) or show a rather diverse set of items (*DIV variant*). Moreover, we want to investigate if this approach can enhance the user experience. The concept behind and the implementation and evaluation of the application have been published in [Lamche et al., 2014c].

4.2.1 Using Conversation-Based Active Learning

Existing Active Learning systems rely on building user models over time, requiring ratings on so-called training points first before making accurate suggestions. However, conversa-

tional systems can show a general set of recommendations right away, based on some initial preferences and try to refine those by asking for feedback up to the desired product. It is unlikely that users always wish to buy the same set of items in our shopping scenario. At some point the user will look for trousers, the next time for shirts or the user might be on a tight budget one day, the other day have more cash to spend available. A conversation-based system adapts quicker to the user's context and interests than systems based on static profiles. Furthermore, the system should not have to rely on ratings from other users to infer recommendations as is done in collaborative filtering systems. Collecting those ratings is time and resource intensive. A content-based approach, more specifically a case-based recommender system does not require any large amount of pre-existing ratings. Case-based systems by design also list products in a uniform manner with a fixed set of properties, which easily enables comparison, beneficial for selecting diverse clothing items. Following that, the search for a suitable algorithm for our shopping scenario was narrowed down to conversation-based Active Learning systems, which would enable quick adaption to the user's current interests.

Progress Modification

The algorithm for the system in this work should be able to show diverse items to enable exploration, but also be able to focus on a particular item region exploiting the user's current preferences, as soon as the user knows what type of item she is looking for (this strategy is later referred to as *DIV* variant). McGinty and Smyth introduced the foundation of such an algorithm called *Adaptive Selection* (see *subsection 4.1.1*) [McGinty and Smyth, 2003b]. However, to allow the presentation of recommendations right from the beginning, without requiring the user to specify initial preferences, we had to modify the algorithm. The initial set of items we show is chosen using *bounded greedy* (see *subsection 4.1.1*) selection to show an as diverse set of items as possible in the beginning to give the user many choices for exploration.

After the system generated the initial set of recommended items, the user has the chance to criticize these items. The approach of McGinty and Smyth does not support the exclusion of a specific item feature [McGinty and Smyth, 2003b]. We therefore extended the algorithm to allow the user to either *like* or *dislike* a specific feature. Due to the limited screen size, our system is using a two-step critiquing process as feedback method (*figure 4.1* shows how the user can influence the calculation of a new set of recommendations by critiquing):

1. If the user has already found a satisfying item, she can select it and the recommendation session will complete. If the user likes some of the item characteristics, she can select the *like* button. Then the whole item is considered satisfactory because the user can either *like* or *dislike* an item and no combination is possible. If most of the item characteristics are unsatisfying, she can select the *dislike* button and the overall item is considered unsatisfactory.

2. In a following screen, the user then specifies which features she either likes or dislikes. As soon as those features are selected, a new set of recommendations is calculated (the corresponding user interface can be seen in *figure 4.7*).

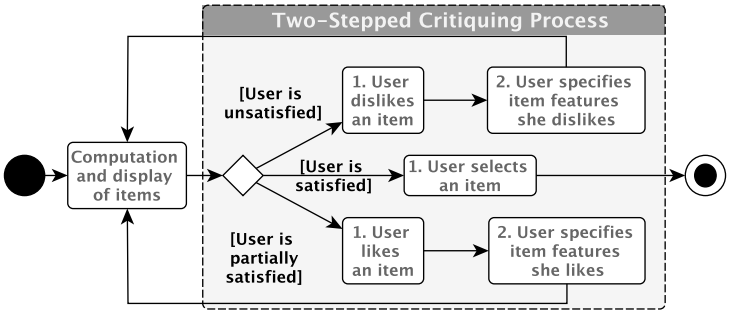


Figure 4.1: The two-step critiquing process

The activity "*computation and display of items*" of *figure 4.1* proceeds as follows: Based on the user's feedback, the system determines whether positive or negative progress has been made. In the first case, if a user dislikes any of the new recommended cases, *negative progress* has been made. Consequently, the system then needs to *refocus* on another item region because current recommendations are not satisfactory. So the next step is to use the *refocus* algorithm and show a set of more diverse items by using the *bounded greedy selection* (see *subsection 4.1.1*). In our application $k = 8$ (amount of results) because the applications presents items in a 3 by 3 grid view and one spot is reserved for the recently criticized item (see *figure 4.7* for an illustration). Consequently, $k = 9$ in the first run where no critique has been performed yet. In this case, the system shows just a very diverse set of items (by using the *diversity-enhancing* strategy) to give the user many choices for exploration. Regarding the *bound* parameter, we obtained best results with $b = 10$. In the third case, if the user liked any of the new recommendations, more similar items are shown (for this purpose, the *refine* algorithm is used). The recommendation algorithm will then show the k most similar items for the next cycle by sorting the case base according to similarity to the query. *Figure 4.2* explains these two different approaches to calculate a new set of recommendations.

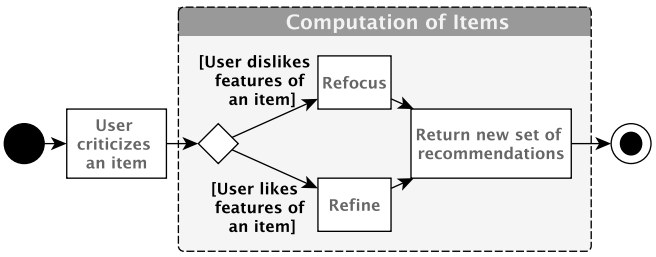


Figure 4.2: The recommendation approach

In our approach, the new set of recommendations also includes the previously criticized item to allow the user to further criticize it for better recommendations. The user might have liked some feature, but then may want to refocus results by excluding features she does not like about this item. If it would not be carried over there is a chance the item is not selected again, potentially robbing the user of her item of choice if the new set of recommendations is worse. However, in contrast to the proposed recommendation approach in [McGinty and Smyth, 2003b], our algorithm reacts not differently if the criticized item in the current cycle has already been criticized by the user. This leads to a more predictable system reaction for the user upon her critiques.

As soon as the user has already performed a critique, the system first modifies the search query q as follows: To query, Ricci and Nguyen presented a logical query with fixed requirements (the user's *must conditions*), favorite patterns (the user's *wished conditions*) and feature importance weights for prioritizing the importance of fulfilling wished features. This will modify the up to now case-based retrieval system of *Adaptive Selection* into a utility-based one (see *section 2.1.4*). However, this approach only allows influencing weights between two features (e.g., *price* is more important than *brand*) [Ricci and Nguyen, 2007]. We therefore developed a solution that enables the user to criticize an item feature as precise as possible. We introduced a weight vector for each feature allowing prioritizing one feature value over the other. This means that our approach allows influencing weights between two features (e.g., if we consider the feature *color*, *red* items can be higher weighted than *blue* items). Features also include context attributes such as the distance to the shop and the current time. For instance the following query q models a user seeking a red dress within a range of 2000 meters and wants to find a shop selling it within the next half an hour:

$$q = ((distance \leq 2000m) \wedge (time\_open = now + 30min)),$$
$$\{color_{red,blue,green}(1.0, 0, 0), type_{blouse,dress,trousers}(0, 1.0, 0)\})$$

To retrieve a set of recommendations for this query q, we use a two-step process similar to [Ricci and Nguyen, 2007]: *First*, nine recommendations are calculated to fill the 3 by 3 grid used in our mobile application (based on the approach presented in *figure 4.2*). *Second*, these nine returned cases are effectively ranked by their similarity to the query q and are presented to the user for feedback (the similarity function will be presented in the following section).

We again adapted the item recommendation method by including previously recommended items in future sets of recommendations. Although McGinty and Reilly argue that the system should prevent users being shown already recommended items [McGinty and Reilly, 2011], initial testing of our system has shown that showing previously recommended clothing items again enables the user to better refine the query.

Determining Similarity

Both, the *refocus* algorithm (based on *bounded greedy* selection), and the *refine* algorithm (based on *query similarity* selection), require a *Sim(q,c)* function that determines the level of similarity in an interval between 0 (no similarity) to 1 (identical) of a query q and an item c or between two items. First a look on how a product is represented as a set of feature vectors: Simple enough each feature has a vector with the value of the product set to weight 1. Continuing with the example from before, a 60 € red dress has the following feature vectors:

$$\begin{aligned} dress_{attributes} = \{&color_{red,blue,green}(1.0, 0, 0), \\ &type_{blouse,dress,trousers}(0, 1.0, 0), \\ &price_{0to50,50to100,100to150,above150}(0, 1.0, 0, 0)\} \end{aligned}$$

To calculate similarity between the query q and any product c or between two products t and c, their feature vectors are fed to a similarity metric $sim_i(t_i, c_i)$. The simplest case are numeric features where a certain value is preferred and higher or lower values are equally less acceptable, for example when looking at the price of an item. For this purpose a symmetric similarity metric like in *equation 4.3* is used.

$$sim_{price}(t_{price}, c_{price}) = 1 - \frac{|t_{price} - c_{price}|}{range(price)} \tag{4.3}$$

The target price t_{price} is compared to the candidate's prices c_{price} by their normalized difference. In this case, the value of the *range* would be the absolute value of the difference between the highest price and the lowest price in the case base.

The final equation to calculate similarity between a query or case t and a product c (with n denoting the number of feature vectors) is based on a weighted sum metric, see *equation 4.4*.

$$Similarity(t, c) = \frac{\sum_{i=1}^{n} w_i * sim_i(t_i, c_i)}{\sum_{i=1}^{n} w_i} \tag{4.4}$$

The result is thus a weighted average of similarity measures of all attributes [Bridge and Ferguson, 2002]. If a feature (e.g., the price of an item) has not been criticized by the user, we assume that it is irrelevant for the user and the similarity algorithm ignores this feature. A simplified version of the similarity metric, where all requirements are equally important (meaning all weights are one), is presented in *algorithm 1.*

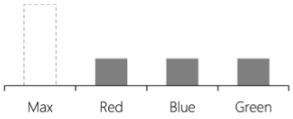
When the recommendation session starts, the set of feature value weight vectors for the query will be empty; the system will only create a vector once a feature has been criticized. When passed to the similarity metric it will ignore any unset feature vectors so processing not criticized features is avoided and the computation is much faster by avoiding useless comparisons.

Algorithm 1 Calculation of similarity

```
1: //Pass query A and item B, or two items A and B
2: procedure SIMILARITY(A, B)
3:     for all projection functions π_f that obtain the values of all defined feature vectors
       f do
4:         if π_f(A) == π_f(B) == 0 then
5:             //Ignore zero feature vectors (e.g.,when feature is not set or has not been
6:             //criticized)
7:             continue
8:         else
9:             sim ← attributeSimilarity(π_f(A), π_f(B))
10:            compared ← compared + 1
11:        end if
12:    end for
13:    //Calculate average
14:    sim ← sim/compared
15:    return sim
16: end procedure
17:
18: //Pass two feature vectors a and b
19: procedure ATTRIBUTESIMILARITY(a, b)
20:    for all value pairs i of feature vectors a and b do
21:        if a[i] == b[i] == 0 then
22:            //Ignore zero weights
23:            continue
24:        else
25:            //Simple symmetric similarity metric
26:            attrSim ← attrSim + (1 − abs(a[i] − b[i])/range)
27:            compared ← compared + 1
28:        end if
29:    end for
30:    //Calculate average
31:    attrSim ← attrSim/compared
32:    return attrSim
33: end procedure
```

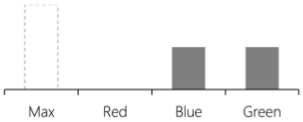
How Critiques Affect Weights

If the user issues a critique on a feature, a new value vector will be initialized for the query with all values being assigned equal weight, for example:



$color_{red,blue,green}(1/3, 1/3, 1/3)$

Then, based on the notion of the critique on a feature value (positive or negative), the system would shift the weights accordingly so a liked value would have the highest weight and a disliked value would be set to a zero weight. Once a feature has been criticized, in consequent cycles, weights are shifted again to reflect the current priority indicated by the user. If the user would like or dislike the color feature (e.g., red) for the first time, this would lead to one of two weight vectors (not actual vectors, numbers chosen for illustrative purposes):



$dislike(red) : color_{red,blue,green}(0, 1/3+1/6, 1/3+1/6)$

By shifting weight in between values instead of just changing single value weights in between cycles the weight sum in a vector is always held constant (at value 1.0). For one reducing other weights evenly makes sure the priority distribution from previous cycles is kept intact as long as possible. This enables the system to still show recommendations, which meet older preferred values if no items matching the currently prioritized value are available. This implicitly prevents the user from specifying conflicting preferences. The latest feedback always overwrites previous ones. If, for example, the color red was liked and at some later point the user decides she dislikes red, the system uses the later dislike critique and conflict resolution is not needed.

Value Similarity Graphs

Right now the algorithm does only prioritize the actual liked value of a feature. For instance, if the user likes that an item is red, the system will only prioritize red with all other colors being equally reduced in weight. Bridge and Ferguson developed a solution to this problem by specifying different similarity metrics for features without any order and those with order. Some feature values (such as the price) can be ordered or some values can be judged as more similar to each other (e.g., red is more similar to purple than to green) [Bridge and Ferguson, 2002]. However, this approach would have been too complex for a mobile recommender system with our requirements and we therefore came up with a more extensible and flexible solution:

Our approach can be called *similarity graphs on feature values*. They are very simple to integrate and understand and work as follows: Every value of a feature is represented as a node in a graph. If two values are deemed similar to each other, they are connected via an edge (see *figure 4.3* for an example). The positive feedback process introduced before is modified slightly to incorporate support for similarity between values. First, the system will look for the node of the currently liked value. Then a list of direct neighbors (distance 1) is built. Now the system will try to uphold the following conditions after the weights have been modified: (a) the actual liked value has the highest weight, (b) similar values do have a lower weight than the liked value, but a higher weight than non-similar values. *Figure 4.3* shows an example graph for the color red. In this example, purple is a similar value, blue is less similar and black is non-similar (so its weight is zero).

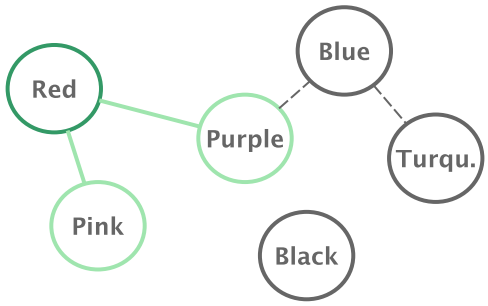


Figure 4.3: A sample of a similarity graph

The Baseline

To successfully test the developed recommendation algorithm, we need to establish a baseline to compare against (later referred as *SIM* variant). This baseline is based on the previously presented *DIV* variant; however, it was chosen to purposely cripple the algorithm by disabling the diversity-enhancing methods. Simply put, when a user criticizes an item, the system first modifies the search query q and then calculates the new set of items by always using the *refine* method (see *figure 4.2*), irrespective of whether the user liked or disliked the item. However, it is reverted back to the original behavior of the *Adaptive Selection* algorithm and never includes previously recommended items. In addition, both algorithms started with a diverse selection of clothing items (identical set of items for both sessions) to reduce bias introduced by the initial selection.

4.2.2 Interaction and Interface Design

After having explained how users can influence the recommendations by critiquing (see *figure 4.1*), we now show how we designed the user interface and interactions to allow the user to easily perform these critiques. Critique actions are represented with *thumbs up* and *thumbs down* icons, known from other popular applications and enabling the user to better predict her action, right below the image and text of each item in the grid. In

a first iteration, the system would display a dialog listing all features of the item that can be criticized (e.g., color, type, price) and asking the user to select one of them (see *figure 4.4*). For example to display more items of the color red, the user would select the thumbs up action for a red item and then select the feature "color". As has been shown in existing research, the user can more quickly move towards her optimal preferences if she can combine multiple critiques in one cycle.

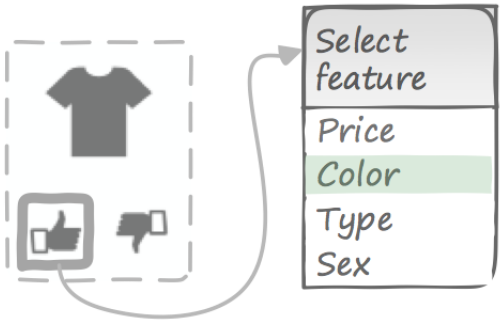


Figure 4.4: The initial interaction design for the two step critiquing process

Another consideration was how to represent the cases recommended by the system. In principle there are two easy ways to use: A list or a grid view (see *figure 4.5*). As mentioned, an important aspect was to represent the items through visual means, images, and to let the text description fade into the background. A list layout would have for example entailed entries with a small image on the left while the larger space would be taken up by a text description (name, price, features). While this would enable to pack more textual information into each item, for the purpose of recommending clothing items, a more image-focused approach seemed best. Therefore a grid view layout was chosen. It enables the display of multiple items in a row, each featuring a large image representing an item, with short additional information in text stuck to the bottom.

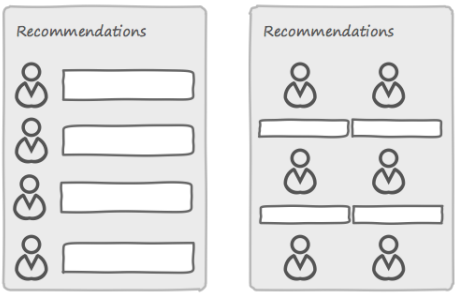


Figure 4.5: On the left a list representation with room for a lot of content next to the image, on the right a grid with less content, but more focus on the images

The implication between a critique and the new set of recommended clothing items remained very transparent to users. Otherwise, this could have resulted in reduced trust into the system and have lowered the probability of users returning to it in consequence. To keep things simple and the interface clean, it was decided to put one explanation text on top

of all items instead of attaching it to each single one, similar to the approach of Shearin and Lieberman (see *figure 4.6*). This allows the user to observe the effect of her critiques and to compare the current profile against the actual displayed items [Shearin and Lieberman, 2001]. It is especially helpful if the system is inferring information, e.g., an exclusion of multiple values of a feature leading to the promotion of the remaining one (e.g., the gender "*male*" after de-prioritizing "*female*" and "*unisex*"). The text lists statements about the priority on values of each feature and can be a combination of three different variants:

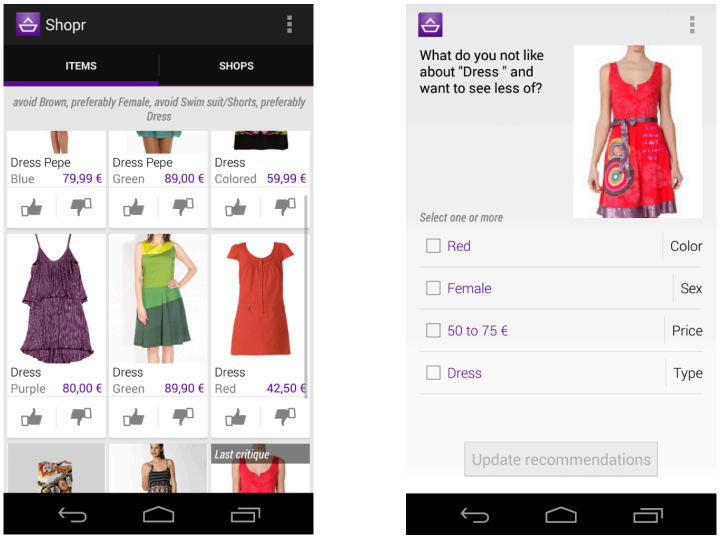
- *Only* <value>. This is shown if one value of a feature gained the maximum weight (e.g., *"only Male"*).
- *Avoid* <value>. This statement is added for each value with no weight (e.g., *"avoid Red"*). To shorten the text, multiple disliked values are being compressed like "*avoid <value1>/<value2>/<value3>*".
- *Preferably* <value>. This positive keyword is added for the value with the currently highest weight (e.g., *"preferably Shirt"*).

These statements were generated for each feature vector, appended and then displayed to the user. The presentation view of the recommendations as well as the critiquing view are illustrated in *figure 4.7*.

Figure 4.6: Example of an explanation text

4.3 User Study

In this user study, we investigate if our developed mobile shopping recommender system based on critiquing instead of queries is appealing to the users and whether a diversity-based or a similarity-based Active Learning approach supports a better user experience. We therefore measure the decision accuracy, the decision effort and the intention to return. In order to estimate the acceptance and usefulness of explaining the current selection of recommended products, we also ask the participants how they evaluate the explanation benefits. Finally, we also want to find out whether people like the general design and usability of the application and ask for overall feedback.



(a) Set of recommendations (b) Critiquing view of an item

Figure 4.7: The final user interface

4.3.1 The Testing Procedure

The test hardware developed for this purpose is a 4.65 inch 720x1280 resolution Android smartphone (Samsung Galaxy Nexus I92503) running versions 4.2.2 and 4.3 of the Android operating system. The *dataset* used for this study was extracted from the now deprecated Google Search API for Shopping [Google, 2013]. Items were associated with those following: An id, one of 13 types of clothing, one of 15 colors, the price (in Euro), the gender (male, female or unisex) and the link to an image of the item. To generate the dataset, the Shopping Search API was queried for keywords associated with types of clothing (e.g., *simply dress*) without any adjectives, to avoid leaning into a particular style as much as possible. This raw data was reviewed by hand to fill in missing and correct wrong property values populated by the API tool, also purging non-relevant items (e.g., wrong type or broken image link). The resulting set is 694 items strong, with 309 for male and 377 for female clothing items. For each clothing type there are between 20 and 130 items, with the majority at around 40 to 50 items per type.

The *testing framework* used to evaluate the critiquing recommender system follows the one presented by Chen and Pu [Chen and Pu, 2009]. The measured data is divided into four areas: *Decision accuracy, decision effort, user intentions and explanation benefits*. More precisely, we measured the following data: *Perceived accuracy, objective accuracy, perceived effort, intention to return, number of critiquing cycles, time consumption* and *explanation benefits*. We measure *explanation benefits* so that we get a sense about the acceptance and usefulness of explaining the current selection of recommended products. We therefore ask testers about any benefits of the explanation text shown and if it in any way made their choice more difficult, e.g., by confusing them with contradicting statements. All perceptive

measures are gathered through a questionnaire listing statements that have to be rated on a 5-point Likert scale (from 1, strongly disagree to 5, strongly agree with 3 being neutral). See *section 2.3* for more details about the evaluation technique.

For the study *participants* of various age, educational background and current profession were looked for. To keep the number of testers at a reasonable size the study was designed as within-subjects, one group of people will test both variants. The order of which system is tested first is alternated between subjects. Also, based on the within-subject design and a big enough number of samples the results can be evaluated using a paired t-test. Overall a number of 28 people participated.The average age was 30, with a maximum of 53. Almost 90% of participants indicated they had online shopping experience (e.g., Amazon or eBay), while only a quarter said they also shop from their mobile devices (for example the Amazon mobile applications or through a web browser, including phones as well as tablets). Two thirds indicated a generally positive attitude towards shopping or browsing for clothing items, whether online or in a store.

To begin with, participants were asked to fill out the demographic part of the survey. Next, the idea of the system was introduced and the purpose of the user study made clear. After introducing them to the task ("find a product you would purchase if given the opportunity") users were given hands on time to familiarize them with the user interface and allow them to grasp how the application works. The full set of questions can be found in detail in *Appendix A*. Which of the two systems was tested first was flipped in between subjects, so a bias because of a learning effect could be reduced as much as possible in addition to the initial hands-on time. After selecting and confirming the choice for a product the task was completed. Now testers were asked to rate statements about accuracy, effort and more based on their experience with the system and to offer any general feedback and observations.

4.3.2 Results

The means of the measured values for the most important metrics of the two systems are shown in *Table 4.1*. Next to the mean the standard deviation is shown, the last column denoting the p-value of a one-tail paired t-test at a significance level of 0.05 (results that are statistically significant are printed bold).

Accuracy

Regarding accuracy, both systems perform well. SIM is seen as more capable of quickly homing in on the area of items that best match a user's preferences. We measured objective accuracy by showing a list of alternative items with similar attributes at the end of each session. When asking if they prefer any of those alternatives, 89.3% of participants stuck with their choice when using the solely similarity-based retrieval approach (SIM), compared to the diversity-enhanced system (DIV) with which 78.6% found their preferred choice.

	SIM		DIV		p value
	mean	stdev	mean	stdev	
Objective accuracy	89.3 %	0.31	78.6 %	0.41	.13
Perceived accuracy	3.71	0.90	3.93	0.90	.12
Critiquing cycles	7.32	7.82	7.50	8.88	.43
Time consumption	216 s	165	201 s	173	.32
Perceived effort	3.39	1.33	3.38	1.20	.46
Return intention	3.32	1.22	3.59	1.12	**.04**
Explanation benefit	2.93	1.39	3.04	1.35	.30

Table 4.1: The means of some important measured values comparing both variations of the algorithm

However, this picture changes when looking at which system participants actually perceived to deliver the best choice out of all available items. Here, the diversity-enhanced approach did achieve better ratings (3.93 compared to 3.71 for SIM, on a 5-point Likert scale). When looking at their actual distribution (see *figure 4.8*) this approach collected twice the number of very positive (value 5 on the Likert scale) ratings and half the number of negative ratings (value 2) compared to the similarity-based approach.

According to the informal statements and observations DIV shows a too diverse set of items when submitting a negative critique on a feature, throwing users a step backwards. Although, those diverse alternatives are in general preferred over the very similar items of SIM improving the exploration capability of the system. It has also to be noted that the overall share of positive ratings (values 4-5) for both variants is high at 71%, with nobody submitting a very negative rating (value 1).

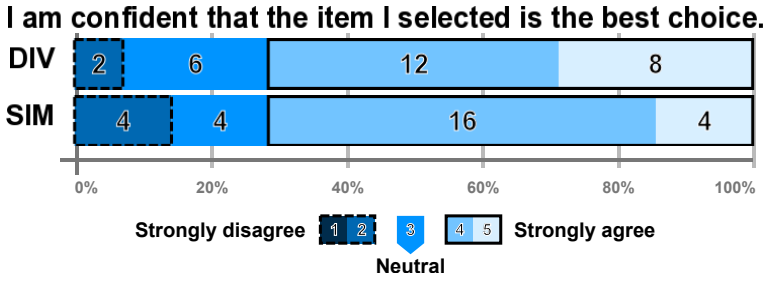
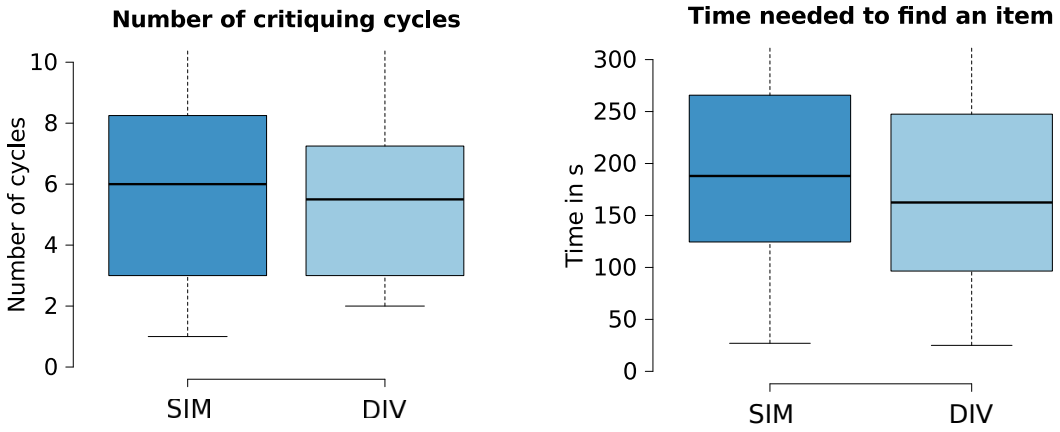


Figure 4.8: Distribution of ratings for perceived accuracy on a 5-point Likert scale

Effort

Comparing the average number of critiquing cycles for each variant, there is only a subtle, non-significant difference. Participants completed their session on average in 7.32 cycles with SIM and in 7.50 cycles with DIV. It was shown for both variants that sessions could

be as short as three cycles regardless the omission of initial preference elicitation. However, looking at the distribution in *figure 4.9 a*, DIV can keep the majority of the number of cycles below 7.25, one whole cycle less than SIM (third quartile at 8.25). In addition, most sessions for both systems are at least three cycles long with DIV achieving a lower median cycle count (5.5 against 6 for SIM). Consequently, the variation in the number of cycles is lower for DIV.



(a) Number of critiquing cycles to find an item

(b) Required time to find an item

Figure 4.9: Box plots of the number of critiquing cycles (left) and time in seconds required to complete a session (the maximum has been omitted for space reasons)

Session completion times are also lower for DIV and typically lie between 1.5 and 4 minutes. On average DIV did beat SIM with a session length of 201 seconds against 216 seconds. This amount of time is acceptable for the mobile usage scenario, where phone use should be kept at a minimum, not forcing users to e.g., look for a place to sit down or safely stand for prolonged periods of time. It also is an acceptable amount of effort compared to visiting a store by chance. Looking at the actual distribution of session lengths *figure 4.9 b* paints a clearer picture, with DIV leading SIM by 25.5 seconds on the median time to complete (162,5 versus 188 seconds). Also sessions with DIV are typically as short as 96.5 seconds (first quartile), where SIM sessions are at least 28 seconds longer (124.5 seconds). On a side note, for both systems a session can be finished in as fast as under half a minute or last up to around 15 minutes (minimum/maximum completion time).

When asked about the ease of finding information and effort required to use the system, the participant's average rating was very similar for both systems (SIM was rated 3.39 and DIV 3.38 out of 5). This is mirrored in the actual distribution of the ratings (see *figure 4.10*), where both variants have similar shares of overall positive (values 4-5, both 51.8%) and overall negative (values 1-2, 25.0% for SIM, 28.6% for DIV) ratings. SIM turns out to be a hit or miss in finding a target item with a low amount of effort. As mentioned, it does not reduce focus when issuing negative critiques like DIV, which contributes a share of positive ratings for little effort. However, SIM is worse in recovering from focusing on the

wrong area of the item space, sometimes requiring more effort from the user for correction. Moreover, the user study showed that SIM sometimes hides potential favorite items or displays unexpected results due to item exhaustion. As DIV shares the similarity-based retrieval on positive critiques with the included option to resurface previous recommendation, it does not suffer from these problems. However, the too broad *refocusing* attempt on negative critiques increased perceived effort again by a little.

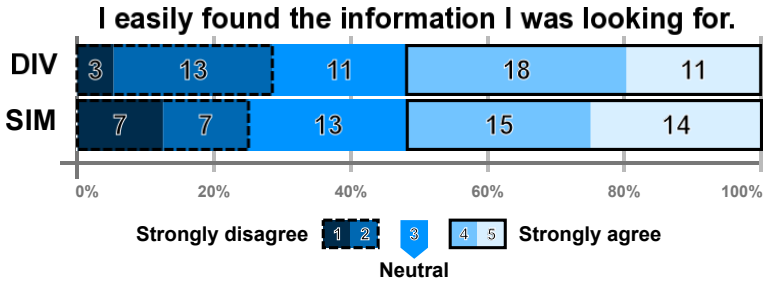


Figure 4.10: Distribution of ratings for perceived effort on a 5-point Likert scale

User intentions

More participants preferred using DIV to SIM and significantly more participants would use the DIV variant again if they needed to look for clothing items in the future (SIM at 3.32, DIV at 3.59 which rated significantly better on a level of $p<0.05$). This is confirmed in the distribution of ratings (see *figure 4.11*), where DIV scores a higher share of overall positive (58.9% compared to 51.8% for SIM) and very positive (25.0% against 17.9% for SIM) ratings. Whereas SIM even gets very negative (7.1%) and a higher share of overall negative (30.4% versus 25.0% for DIV) ratings.

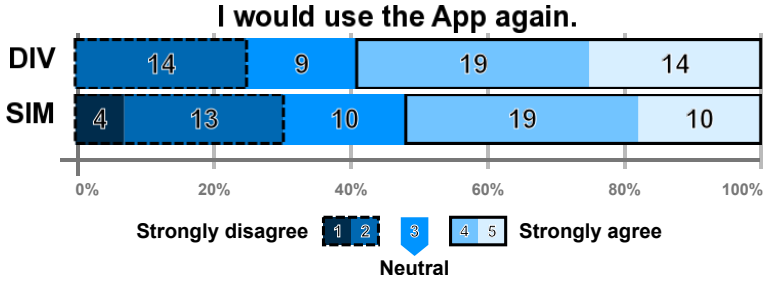


Figure 4.11: Distribution of ratings for the intention to return to the system on a 5-point Likert scale

Overall, the users were very satisfied with this critiquing-based approach instead of a mobile recommender system based on search queries. We therefore proved the applicability of a conversation-based Active Learning recommender system on a modern smartphone platform. With some small changes, like increasing the number of recommendations and

more opportunities for critiquing other features, the application could be made even more appealing to users.

Explanation benefits

Finally, the participants were also asked about the usefulness of the included explanation text. First of all there is no significant difference between the two variants, as can be expected because the functionality was not dependent on the variant used (mean rating SIM 2.93, mean rating DIV 3.04. More importantly the overall number of positive ratings was very high at about 70% with about half of those being very positive ratings. However, as the distribution of ratings shows (see *figure 4.12*) this is somewhat dampened by a sizable chunk of very negative ratings (12.5%).

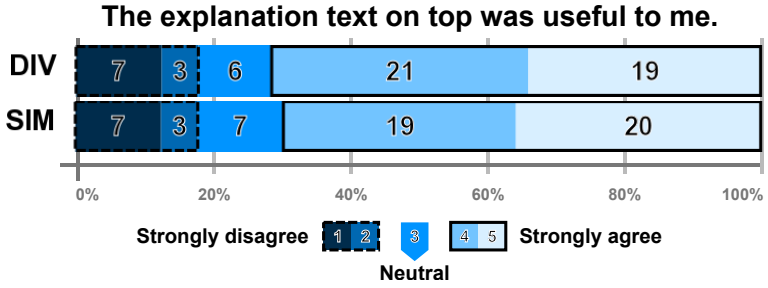


Figure 4.12: Distribution of ratings for the explanation text of the system on a 5-point Likert scale

4.4 Conclusion and Next Steps

In this chapter, we developed a mobile conversation-based Active Learning recommender system for an explorative shopping context using utility-based retrieval and critiquing for feedback. Within a two-step critiquing process, the user can specify if the focus of the recommendations currently is on her desired region of the item space. This feedback process indicates if positive or negative progress has been made according to the last critique. If current recommendations are not satisfactory, negative progress has been made. In this case, our approach augments similarity-based item retrieval by diversity-enhancements through *bounded greedy* selection. The system introduces so-called feature value weight vectors to model a user's current preferences used by the similarity metric to select new recommendations. A mobile Android application using the system was evaluated and produced good results. It performed well regarding accuracy and effort against a purely similarity-based approach, with participants indicating to return to the system for future use. Moreover, the overall feedback of the testers for a critiquing-based mobile recommender system was positive. Early results from a follow-up study show that tweaking the level of diversity in recommendations can positively benefit the perception of the system and therefore may lead to even better results. Right now, we have already developed

a suitable recommendation algorithm for a critiquing-based recommender system but we have not considered the *cold-start* problem so far, where without any personal information the system is unable to provide personalized recommendations. Due to the fact that the performance of a recommender system depends on the system's knowledge about the user preferences, the system performs poorly when it has little knowledge about new users. To allow a positive user experience even for first-time users, a solution to quickly develop a user model during the very first recommendation session has to be found. The following chapter will therefore focus on the user model and present a solution to classify first-time users of a mobile recommender system using Active Learning in a quick and user-friendly way so that the system can generate personalized recommendations right away.

5

User Modeling

This chapter discusses the cold-start problem in a mobile recommender system and presents a solution to generate a user model right from the start. We will come up with a solution based on stereotypes that delivers fast and accurate personalized recommendations. We propose a critiquing shopping application which first uses navigation by asking to determine the user's stereotype to deliver personalized recommendations right away. In a second step, navigation by proposing supports critiquing in a finer granularity level to refine the user model. In a following user study, we will test if this approach is appropriate for a critiquing-based mobile recommender system. The chapter is divided into four sections. It starts by summarizing the state of research on user modeling in recommender systems and covers previous work with special relevance in the areas of stereotype user modeling (1). The second section describes the concept of the user model, the analysis and technical design of the system, as well as some implementation details of the prototype (2). Once the prototype is developed, it is evaluated in form of a user study. The goals of the evaluation, the methodology employed and the results are described in section (3). Finally, the results of the user study are discussed, a conclusion is given and future research is introduced (4).

5.1 Motivation

In this chapter, we investigate, if stereotypes are a suitable approach in a mobile recommender system for a fast and accurate user modeling. Besides a suitable recommendation algorithm, the modeling of the user is indispensable to deliver accurate personalized recommendations. Stereotype user modeling was one of the earliest approaches to user modeling and personalization in general [Rich, 1979b]. A stereotype-based system maps the individual features for the recommendation process to one of several equivalence classes, whose profiles are then used for computing the recommendations. Examining related research, most mobile recommender systems do not explicitly state the user model used behind their recommendation algorithm (e.g., [Gavalas and Kenteris, 2011]). It may be simple or im-

plicitly part of the recommendation algorithm. In order to provide personalized mobile recommendations even in the cold start phase, a user model based on stereotypes seems to be a suitable approach. Most people can be associated with a specific fashion style that barely changes (e.g., *sporty* vs. *gothic*), so that stereotypes can be easily predefined and an already existing user data base is not required. Moreover, the use of a stereotypical user model allows for a quick characterization of users, particularly important for a mobile scenario. So far, no research was found which tried to combine a stereotypical user model with a recommender system on a mobile device. We will therefore examine the effectiveness of a mobile recommender system with a user model based on stereotypes. The main goal of this chapter is to examine whether a stereotype user model leads to better recommendations as part of a mobile recommender system.

5.1.1 Stereotype-Based Recommender Systems

To understand the concept and aim of stereotype-based recommender systems we will explain the fundamentals of stereotype user modeling and give an overview of existing approaches in the literature.

Stereotype User Modeling

User models are a distinctive feature of user adaptive-software systems. A user model represents information about an individual user that is needed to provide an adaption effect of the system, i.e., showing different search results or links while navigating for each user. The user model is built by implicitly observing user behavior or explicitly requesting information about the user. This process is known as user modeling. User modeling directly influences the ability of the system to adapt correctly to each individual user. The more accurate information is available about a user, the better the system is able to adapt correctly [Brusilovsky and Millán, 2007]. A distinction can be made between stereotype user modeling, which has been prevalent in the early days of recommender systems and feature-based user modeling, which is most often used nowadays [Brusilovsky and Millán, 2007]. Derived from Sleeman and Brusilovsky et al., user models are analyzed along three layers [Sleeman, 1985], [Brusilovsky and Millán, 2007]:

- What is being modeled (nature of the model)
- How the information is represented (structure of the model)
- How different kinds of models are maintained (user modeling approaches)

The nature of the model can be subdivided into features of the user as an individual and the context that the user is currently experiencing. The latter is especially relevant for all mobile and ubiquitous adaptive systems. The five most popular and useful features of a

user for user modeling are the *user's knowledge, interests, goals, background* and *individual traits.*

Different forms of user models exist. Bayesian networks are a popular approach to model uncertainty for overlay models. They are a graphical approach consisting of nodes and links whereby each node represents a variable and each link a causality relation with a certain probability. To build a Bayesian network, the qualitative model has to be built first. It represents the domain that needs to be modeled. Next, the quantitative model consisting of the probability distributions for the nodes is constructed [Brusilovsky and Millán, 2007]. *Keyword user profiles* usually extract several keyword vectors from a specific source (e.g., from the browser history of the user) using different weighting schemes or algorithms. The user's explicit and implicit feedback is used in order to build the user profile. *Semantic network user profiles* are built by extracting keywords from user-rated pages which are then added to a network of nodes. Based on the user's explicit feedback on items, the structure of the network is continuously improved. *Concept profiles* are constructed based on training examples of already existing mapping between vocabulary and concepts and are represented by conceptual nodes. These nodes represent abstract topics the user might be interested in, instead of specific keywords. Implicit feedback of the user is used to build the user profile [Gauch et al., 2007].

Stereotype user modeling is an alternative to the nowadays prevalent method of feature-based user modeling [Brusilovsky and Millán, 2007]. A stereotype-based system considers the individual features for the recommendation process and based on those matches them to one of several groups, called stereotypes. Depending on the determined stereotype, personalized recommendations can be generated. Stereotypes are usually organized in a directed acyclic graph to allow for generalizations. *Figure 5.1* shows a generalization of different religious groups in the stereotype "religious person". Each stereotype corresponds to a certain set of features characteristics. If the characteristics of users change they may be reassigned to a different stereotype. In order to match a stereotype to a person, the system needs to have specific *triggers* - events that signal the appropriateness of a particular stereotype and in turn activate it. For one person, several stereotypes can be active. Once activated, the characteristics of the stereotype are incorporated into the user model [Rich, 1979b].

5.1.2 Existing Approaches

Rich explored the use of stereotypes for recommending books in a system called *Grundy* in 1979. A stereotype in *Grundy* consists of a collection of facets which are characteristics of a person with a certain value. The facets themselves are determined by the type of system. Each facet is also equipped with a rating indicating the certainty of the specific characteristic for the stereotype. Thereby stereotypes become a collection of facet-value-rating combinations. As the user model is based on uncertain information, stereotypes are equipped with two more attributes: A rating of the confidence of the prediction, as well as a justification about the prediction [Rich, 1979b], [Rich, 1979a] . A stereotype user

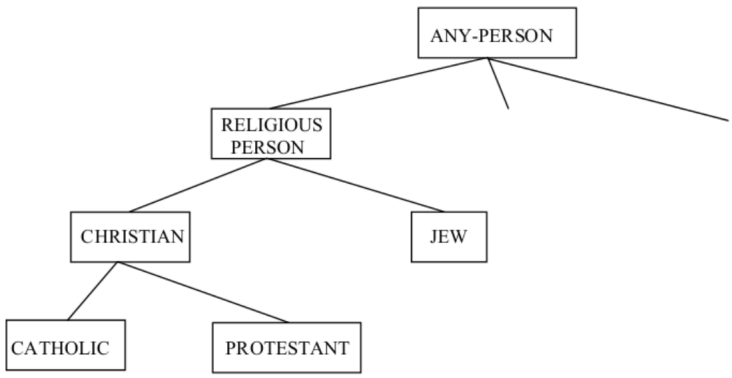


Figure 5.1: A sample of a directed acyclic graph of stereotypes [Rich, 1979b]

model in *Grundy* is thereby made of four attributes: Facet, value, rating and justification. The resulting user model (see *table 5.1*) is a function of the activated stereotypes and their facets together with the level of certainty attributed to them. Some of the facets of different stereotypes may contradict or enforce each other. These situations have to have an influence on the certainty indicated by the rating of the facet. In order to provide recommendations to the user, *Grundy* first collects a set of facets which have a fairly high rating and a value that is not close to the average value. It then iterates over each facet, considering all the books which match the current facet. In the end some books remain and the user model is used again to explain the main features of the book which could be relevant to the user. *Grundy* also has the ability to adapt stereotypes and triggers, if it turns out that predictions based on them are incorrect [Rich, 1979b], [Rich, 1983].

Chin describes an explicit user modeling component system named *KNOME* which is part of the natural language computer consultant system (UC) and assists a user in learning how to use a UNIX operating system. It uses what Chin calls double stereotypes. The system hereby uses the level of the user (novice, beginner, intermediate, expert) as one stereotype and the level of difficulty of the information to be learned by the user (simple, mundane, complex) as another one. It also modeled the individual progress of users to avoid presenting the same lesson twice and a fixed number of rating levels to express the level of certainty about the stereotype assignment [Chin, 1989].

Finin describes *GUMS* (General User Modeling Shell), a system that tries to build domain-independent user models. Instead of having assumptions about the user, the system stores facts about the user which it receives from an application system and tries to verify them. It then informs the application system about potential inconsistencies and answers queries about its current assumptions. The shell allows for the definition of a stereotype hierarchy in form of a tree, but only one stereotype can be applied at a time. The stereotypes contain facts about the user. If contradicted, the system will switch to another stereotype. The system also distinguishes between definite and default parts of the stereotype. While definite parts must apply to all users in the class, the default parts are only initial beliefs which can be changed over time [Finin, 1989].

Facet	Value	Rating	Justifications
Gender	female	1000	Inference-female name
			WOMAN
Nationality	USA	100	ANY-PERSON
Education	5	900	INTELLECTUAL
Seriousness	5	800	INTELLECTUAL
Piety	-3	423	WOMAN
			FEMINIST
			INTELLECTUAL
Politics	liberal	910	FEMINIST
			INTELLECTUAL
Tolerate-sex	5	700	FEMINIST
Tolerate-violence	-5	597	WOMAN
Tolerate-suffering	-5	597	WOMAN
Sex-open	5	960	FEMINIST
			INTELLECTUAL
Personalities	4	646	WOMAN

Table 5.1: A sample of a user model from *Grundy* [Rich, 1979b]

The authors Ambrosini et al. used a new approach to stereotype user modeling by generating stereotypes with the help of case based reasoning and an artificial neural network. Thereby old cases of the recommendation domain are used as the input to the neural network. The result is a set of stereotypes which can be used as the new input to the neural network, which is thereby built up over time [Ambrosini et al., 1997].

Ardissono and Goy use stereotypes for a web-based virtual market for telecommunication products called *SETA*. The system asks the user to initially fill out a form providing information about, among others, the user's age, gender, job, and education level. Based on the provided information the user is categorized among different stereotype families: Domain expertise, life style, graphical interface requirements (color, background, font size) and desired type of use of acquired goods (business vs. private usage). Based on the stereotypes, predictions are made on the user's features including receptivity to information, sight (capability to read small texts), domain expertise, technical interest, aesthetic interests as well as predictions on preferences for product properties such as quality, ease of use, portability, technicality, cost, novelty and design. Depending on the model of the user, items are sorted by highest compliance with the user's preferences. The system includes a dynamic user modeling component which monitors user behavior and adapts the user model accordingly. The products are described according to a knowledge base and are categorized by properties which correspond to the mentioned user preferences. The match of a product to a user model is based on a multiplicative formula to better deal with contradictory information. The authors comment that the models for users and products are highly specialized on the domain, in this case telecommunication. If no sufficient data

is available about the domain, the authors recommend to build a system which automatically learns the stereotypes [Ardissono et al., 1999], [Ardissono and Goy, 2000]. A similar approach of initially assigning a stereotype and refining the user model as time goes by is also implemented in [Virvou and Tsiriga, 2003] and [Ardissono et al., 2001].

The authors Micarelli and Sciarrone describe an adaptive web-based filtering system which acts as a shell to the search engine Alta Vista. The system uses a user modeling component named HUMOS (hybrid user modeling system) which builds long-term models of internet users and their information needs. The system uses stereotypes together with a semantic network of informative words about the domain. Stereotype assignments are based on artificial neural network. Stereotypes describe the information need in the system and are comprised of various slots that contain a domain, a topic and a weight. The weight indicates the interest in the domain-topic combination. A slot also contains semantic links which are co-occurring terms to the domain and justification links which indicate the source of the weight attribution. Topics build the core of a semantic network of co-occurring terms. The user model is dynamic, i.e., it adapts the stereotypes over time. The evolution is based on feedback and direct editing. A TMS (Truth Maintenance System) is used to keep track of dependencies among beliefs and to retract them if necessary. The user modeling process is also based on case based reasoning as described in brief earlier [Micarelli and Sciarrone, 2004].

Comparing the above mentioned stereotype-based recommender systems, it becomes clear that so far, no research tried to combine a stereotypical user model with a critiquing recommender system. While stereotype user models were some of the first user models ever implemented in systems with a recommending purpose, their potential value for a critiquing recommender system has not yet been evaluated. As presented in *chapter 4*, critiquing recommender systems allow the user to criticize the suggested items at every recommendation cycle and have proven as an effective approach to elicit the user's preferences and thus to improve personalized recommendations. By eliciting critiques, the system narrows down the user's interests until the desired item is obtained. We hypothesize that involving the user in a conversation is a suitable approach to determine the user's stereotype quickly and to deliver personalized recommendations right from the start. We argue that especially in an exploratory scenario such as going shopping without having a specific item in mind or looking for a restaurant during a city trip, the user should get personalized recommendations as soon as possible without having to create a detailed user profile.

5.1.3 Goals

The scenario in mind is that of a critiquing fashion recommender system. In particular in the fashion domain, most people can be associated with a specific fashion style that barely changes (e.g., *classy* vs. *hipster*), so that stereotypes can be easily predefined and an already existing user data base is not required. As soon as a user starts the system, it should recommend items suitable to the user's preferences and provide detailed infor-

mation. The application should be designed for a modern smartphone platform to allow recommendations for a mobile scenario. Any user familiar with a smartphone application should be able to use it without problems. The main goal of the prototype is to examine whether a stereotype user model leads to better recommendations as part of a critiquing recommender system. The system should be able to initially build up a user profile based on stereotypes and consequently enable critiques to improve the recommendations. Our system therefore combines two critiquing approaches (see *section 2.2* for a detailed description) [McGinty and Smyth, 2003a]. On the one hand *navigation by asking* engages the user in a dialogue where the user is asked to answer questions about preferred features of an item. Our approach uses this technique to quickly determine the user's stereotype. Here, the difficulty is to select few revealing questions so that the user will not be bothered and simultaneously the stereotype accurately determined. Otherwise, *navigation by proposing* asks the user to provide feedback on concrete items. Once our system issues a new set of recommendations, it will solicit a user critique (e.g., indicating that the suggested item is too expensive) to again issue an improved set. To allow personalized recommendations, the particular user model as well as the user's critiques should be taken into account. In order to evaluate the system, a version without a stereotype user model has to be developed as well. The two different applications will then be compared within a usability test. In conclusion we want to answer the following questions:

Research Questions: Can a stereotype user model improve the recommendations provided by a critique-based recommender system? How to build a stereotype user model in the domain of fashion? How to build up the stereotype user model using the navigation by asking approach without negatively affecting the user experience? How can methods of the critique-based recommender system developed in *chapter 4* be integrated in the user interface and support the recommendation process?

Driven by these demands, this chapter will also answer if a user model based on stereotypes is a good approach for a mobile recommender system.

5.2 Designing the Test Application

The goal of the test application is to investigate if a mobile recommender system based on a stereotype user model delivers more accurate recommendations than one without. Moreover we want to find out if this approach can improve the user experience. The concept, the implementation and the evaluation of the stereotype have been published in [Lamche et al., 2014b].

5.2.1 Stereotype Concept

Until now, there is little academic research on stereotypes for the fashion domain. Therefore research on publicly perceived stereotypes was limited to information found on the world-

wide web, e.g., [Angerosa, 2011]. We compared the most frequently classified stereotypes based on their given definition and finally identified the following ten fashion stereotypes:

- Indie/Hipster: Typical for being original, dressing different than others. Clothes use earth tones, pale pink, or cream colors.
- Emo: Derived from emo (emocore) music. Clothes typically use dark colors like black, brown, grey or neon colors as a contrast.
- Preppy: A person who cares a lot about personal appearance. Clothes are usually rather expensive with bright colors.
- Gothic: Fashion style that is related to gothic rock music. The style is first and foremost known for using black as the main color in contrast to pieces of red, scarlet or deep purple. Clothes are often made of leather or lace.
- Urban: Also known as the style of the street and thereby referred to as street fashion. Typical are denim details, jeans, simple t-shirts and sneakers. The style often contradicts big brands with inexpensive items.
- Athlete/Jock: Mostly a male stereotype. Relates to a person that does a lot of sports or is a big sports fan. Typical clothes are training outfits or outfits that emphasize their physique on a daily basis.
- Skater: Related to the skateboarding subculture. Clothes typically include shirts in all colors with messages, baggy or nowadays often super tight pants, as well as hoodies with big logos.
- Girly: A female stereotype that tries to recreate the image of a young, innocent girl. Clothes are therefore mainly skirts and dresses in bright colors like pink, red, yellow, white or light blue.
- Classy: Fashion style of people mostly above 40 years of age who have established a certain lifestyle combined with a high stable income. Clothes are in the upper price range, mostly plain and feature colors like black, white, blue or red for women.
- Mainstream: The style of the average person. Influenced by various other current fashion styles, but in a less extroverted version. Clothes are typically in a moderate price range.

For the allocation of items to stereotypes we use a *weighted keywords* approach (see *subsection 5.1.1*). Out of the features identified for the various stereotypes, a limited set of attributes consisting of colors, brands and general descriptions for the clothing was identified. Each stereotype has manually been given a rating on a scale of 0 to 10 for each attribute, representing the *weight* to which the feature is related to the stereotype (e.g., the stereotype *girly* gets an attribute score of 9 when an item has a pink color). The complete list of ratings is listed in *Appendix B.1*.

The first screen of the application uses the *navigation by asking* approach (*see subsection 5.1.3*) to determine the users' stereotype. The users are asked to answer four questions about their age, gender, profession and music taste (the music taste is taken into account because studies found out that it is highly related to the individual fashion style [Na and Agnhage, 2013]). The age groups used are: Younger than 13, 13-18, 18-30, 30-50, 50-65 and older than 65 years old. Having answered these questions, the system computes the three most relevant stereotypes based on the elicited information (the two corresponding user interfaces can be seen in *figure 5.3*). Each stereotype has been given a weight for all available age groups, jobs and music styles. The stereotype algorithm iterates through all stereotypes available and adds up the likelihood that this stereotype has for each of the properties age, job and music. The resulting three stereotypes are presented as pictures to the user. We use a picture-based approach because also recent research emphasizes that a non-verbal way of interaction, e.g., by asking the user to select a set of pictures, is experienced as exiting and inspiring [Neidhardt et al., 2014]. The fifth and final question therefore asks the user to select the picture that most likely represents their individual fashion style. The algorithm for the determination of the top three stereotypes is presented in *algorithm 2*. *Figure 5.2* illustrates the user-system interaction process.

Algorithm 2 Calculation of top three stereotypes

```
1:  procedure TopThreeStereotypesDetermination(stereotypeForm)
2:      stereotypeLikelihoodMap ← initilizeNewMap()
3:      //Iterate through all stereotypes available
4:      for all stereotypes do
5:          likelihood ← 0
6:          //Continue if stereotype is valid for gender of user
7:          if genderOfStereotype = genderOfActiveUser then
8:              //Determine likelihood of all user properties for the stereotype
9:              likelihood ← likelihood + getAgeLikelihood(stereotype, stereotypeForm)
10:             likelihood ← likelihood + getJobLikelihood(stereotype, stereotypeForm)
11:             likelihood ← likelihood+getMusicLikelihood(stereotype, stereotypeForm)
12:         end if
13:         addLikelihoodToStereotypeLikelihoodMap(likelihood)
14:     end for
15:     sortedMap ← sortByLikelihood(stereotypeLikelihoodMap)
16:     return getTopThreeStereotypes(sortedMap)
17: end procedure
```

5.2.2 Recommendations and Critiquing

As soon as the user selects the preferred stereotype, stereotype-based recommendations are calculated and shown in a grid view. The view is scrollable until the end so that all recommendations are visible in one screen. The recommendation algorithm sorts the items by their expected interest for the user. It first gets all attributes and their values for the

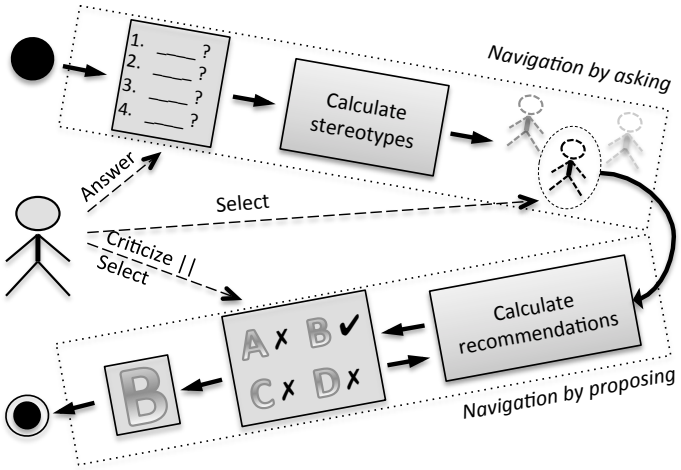
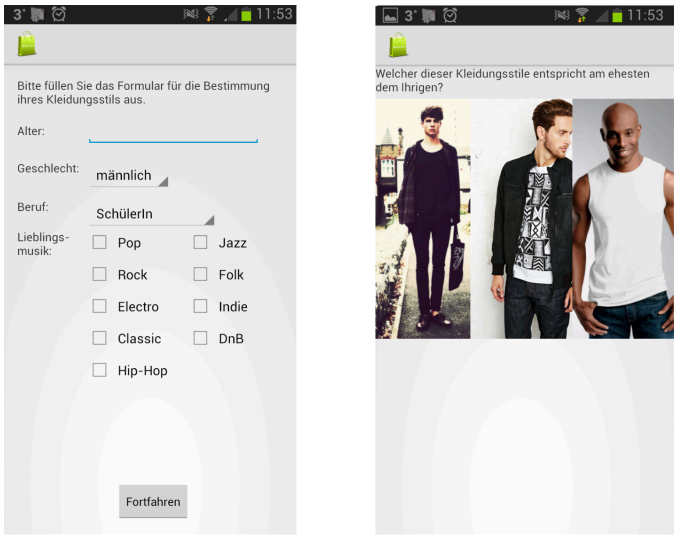


Figure 5.2: The user-system interaction process



(a) Stereotype questionnaire view (b) Stereotype selection view

Figure 5.3: The stereotype determination interfaces

active stereotype and then scans each item for the attributes color, brand and description. If the checked item contains one of the stereotype attributes, the specific *attribute weight* is added to the proximity measure. All weight values for the found attributes are thus added up and then divided by the number of found attributes. The result is a value for each clothing item which indicates the expected interest for a user with the selected stereotype. These values will be sort in descending order and then presented as clothing recommendations to the user (see *figure 5.2*). It is worth noting that we found out during testing that brands provide the most reliable indicator for the attractiveness of a clothing item to a person. The impact of the brand name on the item's ranking therefore takes

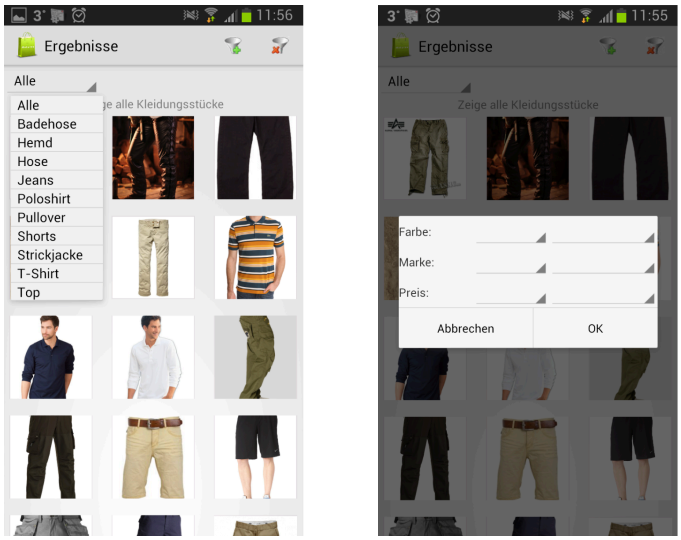
up half of the total impact of all attributes. *Algorithm 3* describes the ranking of clothing items according to the active stereotype.

Algorithm 3 Calculation of recommendations

```
1: procedure ClothingItemRecommendation(stereotype)
2:     clothingItemProximityMap ← createNewClothingItemProximityMap()
3:     for all clothingItems do
4:         proximity ← 0
5:         hits ← 0
6:         //Compute proxmity for all attributes other than brand proximity first
7:         haystack ← getClothingItemAttributes(clothingItem)
8:         //Get all relevant attributes for the active stereotype
9:         attributeProbabilityMap ← getAttributeProbabilityMapOfStereotype()
10:        //Check whether these attributes appear in the item, if so add their weight
11:        //to the proximity measure
12:        for all attributes do
13:            if haystackContainsAttribute() then
14:                //If the needle is found add its weight
15:                weight ← getProbabilityOfAttribute()
16:                proximity ← proximity + weight
17:                hits ← hits + 1
18:            end if
19:        end for
20:        //Depending on number of attribute hits set weight for brand impact
21:        brandImpact ← hits > 2 ? hits/2 : 1
22:        brandProbabilityMap ← getBrandProbabilityMapOfStereotype()
23:        for all brands do
24:            if itemIsOfBrand() then
25:                brandWeight ← getProbabilityOfBrand()
26:                proximity ← proximity + (brandWeight · brandImpact)
27:                hits ← hits + brandImpact
28:            end if
29:        end for
30:        //Divide proximity by the number of all hits for normalization
31:        clothingItemProximity ← hits > 0 ? proximity/hits : 0
32:        addProximityToClothingItemProximityMap(clothingItemProximity)
33:    end for
34:    sortClothingItemProximityMap(clothingItemProximityMap)
35:    return clothingItemProximityMap
36: end procedure
```

An implemented drop-down menu allows for *user-initiated critiques* (see also *figure 5.2*). If the user is not satisfied with a presented item or wants to limit the number of visible items, she can reduce the items to a specific clothing type. When the user clicks on the image in the upper right corner, she can additionally adjust the clothing color, brand

and price range. For example, a user can state that she only wants to see red-colored dresses within the price range of 50 € to 100 €. *Figure 5.4* shows the two corresponding critiquing interfaces. A text field above the results is always visible, listing all the critiques that have already been issued. The last performed critique can be reversed by pressing an *undo* button in case that the user is not satisfied with the remaining items of choice. Several critiques can be set simultaneously and only for attributes that are still present in the currently seen results. Based on these critiques, the system calculates a new set of recommendations. Clicking on an item opens a new screen with a more detailed description. The recommendation session finishes once the user selects a satisfying item.



(a) Adjustment of clothing type (b) Adjustment of color, price and brand

Figure 5.4: The two different critiquing interfaces

5.3 User Study

The main goal of the evaluation is to find out whether personalized recommendations can be improved through a critique-based stereotype determination. This question shall be answered by measuring decision accuracy and decision effort. Another aspect of the evaluation is the overall user experience. We want to examine how users feel about the questioning procedure and if they feel bothered by the navigation by asking approach. Moreover, users get the possibility to criticize the recommendations. The user study should answer the question whether and how users use this critiquing approach or whether they prefer to scroll down until they find a suitable item. At last we want to get feedback on the overall design of the application. All of these questions are to be answered by qualitative questions.

5.3.1 The Testing Procedure

The prototype was written for the Android API version 19 and supports all devices running Android API version 8 or higher. The clothing item *dataset* used for this study was extracted from the now deprecated Google Search API for Shopping [Google, 2013]. The raw information from the API was rather limited with most information having to be extracted from the item and store description. To generate the dataset of clothing items, the Shopping Search API was queried for keywords associated with types of clothing (e.g., simply 'dress') without any adjectives, to avoid leaning into a particular style as much as possible. The dataset built contains 668 different clothing items of 263 different brands. Items were associated with the following features: An id, one of 13 types of clothing, one of 15 colors, the price, the gender, a description and the link to an image of the item.

The *testing framework* used to evaluate the critiquing recommender system follows the one presented in Chen and Pu (see *section 2.3* for more details about the evaluation technique) [Chen and Pu, 2009]. In this user study we measured the following data: *objective accuracy, perceived accuracy, critiquing cycles, time consumption, perceived effort, critiquing convenience* and *user intentions*. To keep the number of testers at a reasonable size the study was designed as *within-subject*, one group of people tested both variants. The first approach is a test with the critiquing recommender system using *navigation by asking* to determine the user's stereotype. The stereotype-based recommendations can then be criticized by the user (*navigation by proposing*). The *baseline* does not perform a stereotype determination. The start screen just presents a diverse item selection which is equal for all users. However, also this approach elicits critiques due to *navigation by proposing*. The complexity of the experiment is kept low by asking users to choose one item they would purchase if given the opportunity for each approach. A potential user bias by using one approach before the other and thereby expecting different results, is reduced by not making the user aware which approach is currently used and randomly switching the order of execution. All variables other than the recommendation algorithm used are kept fix. After having performed the task for each approach, candidates are asked to fill out a demographic questionnaire and to rate statements about the system design, the perceived ease of finding information and effort required to use the system and to perform the critiques, the perceived accuracy of the suggestions, the intention to actually buy the product and reuse the system. Most questions can be answered using a 5-point Likert scale (from 1, strongly disagree to 5, strongly agree), some only have the options of 'yes'/'no'/'do not know' while some questions are open. The full set of questions can be found in all detail in *Appendix B.2*.

For the study, *participants* showing an interest in using the described application were recruited. The user study finished with 32 participants, 27 male and 5 female, with an average age of 28 years, ranging from 22 to 54. The participants were from 6 different professions, most of them being students, software developers or research assistants.

	Stereotype		Baseline		p value
	mean	stdev	mean	stdev	
Objective accuracy	0.47	0.34	0.32	0.34	**.036**
Perceived accuracy	3.5	0.53	2.6	0.52	**<.001**
Critiquing cycles	2.56	2.63	3.76	2.93	**.006**
Time consumption	47.82 s	35.83	64.26 s	33.68	**<.001**
Perceived effort	57.9 %	-	42.1 %	-	-

Table 5.2: A comparison of the user study's results

5.3.2 Results

The data was analyzed using averages, standard deviations and student's t-test for determining distribution differences. A one-tail paired t-test at a significance level of 0.05 was performed to calculate the p-value. *Table 5.2* shows the means for the most important metrics of the two systems, the standard deviation, as well as the p-value (significant results are printed bold).

Accuracy

Objective accuracy can be measured using the *R-Score* which is based on the assumption that the value of a recommendation declines exponentially with the position of an item (see *section 2.3* for more details about its calculation). A higher *R-score* refers to a better ranking of the item. Calculating the *R-score* for the selected items leads to a mean of 0.47 ($\sigma = 0.34$) in stereotype mode and 0.32 ($\sigma = 0.34$) in the baseline. So we conclude that the stereotype-based approach is significantly more accurate at a 0.05 level (p-value = 0.036).

To determine the *perceived accuracy*, users were asked whether they would purchase the item they last selected. The answers to the question were put on a 5-point Likert scale (from 1, strongly disagree to 5, strongly agree). The stereotype iteration was rated significantly better (see *figure 5.5*).

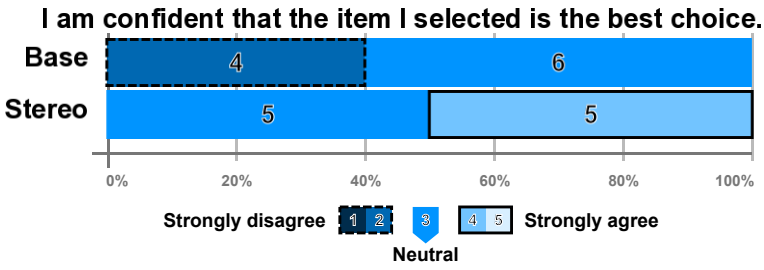
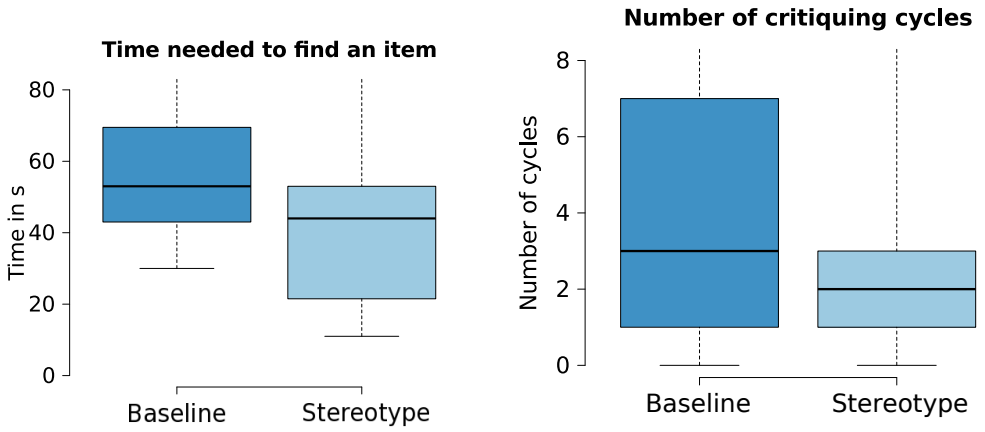


Figure 5.5: Distribution of ratings for perceived accuracy on a 5-point Likert scale

Effort

Objective effort is measured in terms of the time a user needs to find a satisfying item and number of critiquing cycles. On average users took significantly less time to complete the task when supported by a stereotype-based user model, in particular 47.82 seconds versus 64.26 seconds for the baseline (see *figure 5.6a*). The stereotype user model also needed significantly less critiquing cycles (a median of 2.56 versus 3.76 for the baseline approach - see *figure 5.6b*).



(a) Required time to find an item

(b) Number of critiquing cycles to find an item

Figure 5.6: Box plots of the time in seconds (left) required to complete a session and number of critiquing cycles (the maximum has been omitted for space reasons)

Perceived effort refers to the difficulty a subject has during the performance of the task in terms of information processing. 57.9 % of the participants preferred the stereotype round and 42.1% preferred the baseline.

User Experience

Critiquing convenience. Only 6.25% of the users stated that they found the form at the beginning of the application inconvenient, thereby confirming that the effort to reduce the time necessary for determining stereotypes has been successful. 44% of the testers considered three to six questions within the *navigation by asking* approach acceptable, 25% one to two and the others more than 6 questions (*note:* Our stereotype determination process asks five questions). 81% of the participants used the possibility to perform critiques while the others scrolled exclusively. Out of those, 44% also used combined critiques (e.g., criticizing the color as well as the brand). Confusions of users were mainly about the setting of the critiques. Users wanted to set several critiques on one attribute, such as 'no red, no blue color', which was not implemented in the system. Positive feedback has

been received on the clean design and simple usability as well as the general possibility to perform critiques.

User intentions. The participants were overall very satisfied with the design of the application (62% with 28% feeling neutral about it) and 91% understood the usage of the application quickly. 63% of the participants stated that they were interested in the scenario of using an application that recommends clothing items, while 75% stated that they would have used the application tested for the scenario. The reasons of those who would not want to use the application included a too limited set of items, inflexible critiques and a slow processing of clothing item images. 66% of the participants were satisfied with the selected item and felt confident that they would purchase such an item.

5.4 Conclusion and Next Steps

This chapter introduced stereotype user modeling in a critiquing recommender system for an explorative shopping scenario to overcome the cold-start problem. Personalized recommendations are generated by determining the user's fashion stereotype and by eliciting user critiques. 10 fashion stereotypes are identified and included in the user model. The system uses *navigation by asking* to determine the user's stereotype in combination with *navigation by proposing* to offer the user the possibility to criticize stereotype-based recommendations by clothing type, color, brand and price. Finally, a prototype using this concept was developed and evaluated among 32 participants. The goal of the prototype was to provide the means to measure the effectiveness of the recommendations as well as the user experience. We have shown that a system using *navigation by asking* to deliver stereotype-based recommendations right from the start delivers more accurate recommendations and simultaneously means less effort for the user. Additionally, we have shown that our system improves the user experience. Users generally liked the user interface, appreciated the options for performing critiques and the questioning part in order to determine the fashion stereotype was not considered as bothering. So far, we already came up with an effective recommendation algorithm that also takes the user's feedback into account and a solution to generate personalized recommendations for first-time users. However, the design of the prototype is still very simple and we have not evaluated different interaction techniques for a user-friendly mobile recommender system. The next chapter will therefore investigate different user interface designs and interaction techniques for a mobile recommender system to allow a smooth human-system interaction and improve the user experience.

6

Interaction Design

Our goal in this chapter is to investigate how to design the user interaction and usability of a mobile shopping recommender system. Smartphones reveal additional characteristics compared to desktop systems being devices that have smaller screens, a direct touch input method and can collect information about the current environment. For that purpose, we will investigate an interaction design process, involving work on establishing requirements, designing solutions that meet these requirements, producing an interactive prototype of the solution and finally, evaluating it. This chapter consists of four sections. The first section (1) provides a brief overview of related work and introduces important foundations of user interaction design for mobile devices. We then depict the interaction design process and explain the development of our low- and higher-fidelity prototypes (section 2). The following section (3) displays the methodology and results of the conducted two-part evaluation. We close by presenting future directions this research topic could take (4).

6.1 Motivation

In this chapter we present the user interaction design process investigated for a mobile product recommender system. Product recommender systems are web-based tools constructed to ease the process of searching and browsing for items in the broad online space. Besides focusing on accuracy in recommender systems research, the user experience of recommender systems is getting more and more important nowadays. A common factor that supports a smooth user experience includes transparency and control management, while also ensuring that the level of cognitive and interaction effort is kept to a minimum [McGinty and Reilly, 2011]. Previous product recommender systems engaging with the interaction and usability issues have heavily focused on conventional desktop-based environments, as in [Pu et al., 2011b] and mostly ignored product recommender systems running on a smartphone. However, when designing for smartphones, there are three main challenges to face that do not encounter when designing for desktop-based systems.

First of all, user interaction in smartphones takes place by using specific touch gestures. Second, screen capabilities are drastically reduced, offering not much space for information and navigation possibilities. Finally, people use their smartphones in all kinds of places, performing various activities and being interrupted by many environmental and social factors [Tidwell, 2010]. This leads to the conclusion that simply taking a complete web-content and squeezing it into a smartphone screen is a rather undesirable approach. This chapter will therefore depict several interaction and interface designs for a mobile product recommender system on a smartphone. It will also examine the question whether users' interaction preferences stay the same in all circumstances or if they rather change when in a different contextual situation. As application scenario serves a mobile shopping recommender system.

6.1.1 Critique-Based Recommender Systems

Finding a specific item in a large collection of available products can become a demanding task for the user. *Critique-based recommender systems* elicit the user's preferences and suggest items whose attributes match the preferences. These systems are a type of *conversation-based recommender systems* (see *subsection 2.2*) and focus on supporting the user in the process of describing, identifying and selecting a user-tailored product to purchase. Critique-based recommender systems are not just assisting the process of search, but also the process of decision making. Their main task is to provide effective search and navigation mechanism in guiding users to find their preferred products in e-commerce based on explicitly stated preferences.

Interaction Design Process

Designing interactive products requires not only an appropriate mobile-device design, but also an appropriate design considering who is going to use the product, how it is going to be used, and where [Rogers et al., 2011].

Figure 6.1 demonstrates the interaction model of critique-based recommender systems dispensed in three key activities: The interaction starts with the system eliciting the user's preferences regarding the item space. Next, the system filters the space of options and presents a recommendation set to the user according to these preferences. The user can then revise the preference model by giving feedback on an item in the form of "I like this item, but cheaper" (*step 3* of *figure 6.1*) as long as she finds a satisfying item which results in a successful termination of the process (*step 4*) [Pu et al., 2011b]. This style

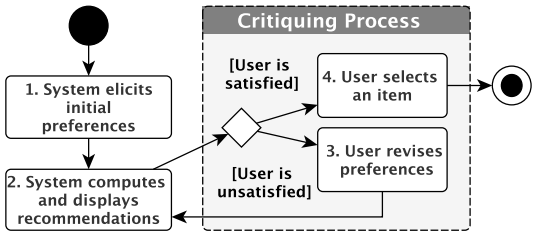


Figure 6.1: The interaction model

of feedback is called *critiques*, and the recommender system class using it *critique-based recommender systems* [McGinty and Reilly, 2011].

The *interaction design process* of critique-based recommender systems involves work on establishing requirements, designing solutions that meet those requirements, producing a (interactive) version of the solution, and evaluating it. These activities inform one another and are repeated, as shown in *figure 6.2*. Evaluating means including users in the design process, e.g., by developing simple or more comprehensive prototypes which can then be tested by a target group [Rogers et al., 2011].

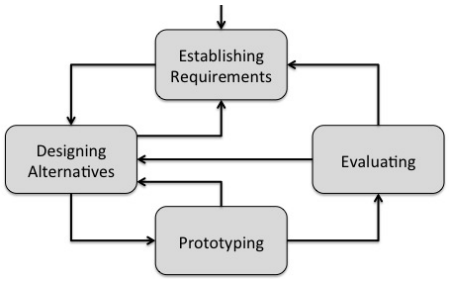


Figure 6.2: The interaction design lifecycle [Rogers et al., 2011, Fig.9.3]

6.1.2 Existing Approaches

Although disciplines regarding interaction design, such as interface, usability or user experience, are essential parts of the recommender system, there are not many sources in the literature engaging in the mobile interaction design for product recommender systems. The majority of researches conducted assume desktop web-based platforms, with critiquing as its feedback strategy. Because of the already mentioned limitations and challenges of mobile systems, only few desktop-based recommender systems are adjusted for mobile use.

McGinty and Reilly delivered a comprehensive outline of previous work on interface considerations across critiquing platforms which focuses on scaling to alternate platforms, manipulation interfaces, explanations, visualization and multi-cultural usability differences [McGinty and Reilly, 2011]. They denote that "*different domain and platform characteristics present recommender interface designers with very different technical and usability challenges*" [McGinty and Reilly, 2011, pp. 438].

Pu et al. established a set of eleven usability-guidelines found on the interaction model. They include, among others, how and in which order to elicit the initial set of preferences, how many and which recommended items to present, what to do in case there are no items in the items space matching the user's preferences, and so on [Pu et al., 2011b].

Each of the described researches did not focus on the mobile environment. It remains unfamiliar how users would interact with the system and perceive subjected values in a context-changing environment, on a device with much-smaller screen sizes and less keypad functionality.

CritiqueShop is an experimental study conducted both in a desktop and mobile environment. The desktop interface has been scaled down to the iPhone. The key considerations influencing design were the limited screen area and direct user manipulation via touch-sensitive user interfaces. Their study demonstrated that users are more likely to apply visual critiques over the textual form, reducing the interaction times of a critiquing

session [Zhang et al., 2008], [McGinty and Reilly, 2011]. *CritiqueShop* was developed to examine critique-based recommender systems on a smartphone, but restricts its feedback strategy to solely critiquing and lacks the context knowledge.

TIP is a mobile system that delivers information about sights (information objects) based on the user's context: Location, travel history and personal profiles describing interest in sight groups and topics. Recommendations are also given based on user feedback and profiles. The paper presents several challenges in the user interface and interaction design. For example, to distinguish between sights that are close and sights that are distant to the user's location and accordingly apply different color schemes [Hinze and Buchanan, 2005].

ReRex is a travel planning iPhone application that recommends situation-adapted points of interests (POIs) to mobile users according to the current context. Its primary goal was to discover whether context data influences user ratings, however, interesting are also its interaction abilities with the user. The application presents the recommendations generated by the predictive model and justifies the recommendations with a direct and simple explanation of the main reason why an item is recommended for that particular contextual situation. A feedback option enables the user to enter her context-dependent rating on the selected POI. Users can change the contextual conditions at any moment. This causes recalculations of recommended items and updates on the suggestion list [Baltrunas et al., 2011], [Baltrunas et al., 2012].

The two mobile recommender systems *TIP* and *ReRex*, dispose a research gap in comparison to this work: The systems rather focus on the development of an accurate application for a tourism's scenario than evaluating different aspects of the user interaction design of mobile product recommender systems.

6.1.3 Goals

Within our research we want to find out which interaction and interface possibilities provide the best usability and user experience in supporting the interaction model steps (see *figure 6.1*). Our second main goal is to conclude whether contextual changes (e.g., of location, budget or weather) imply changes on the users' preferences about their favorite interaction and interface options for a mobile recommender system. The prototype should run on a modern smartphone platform to test the user interaction process in a mobile environment. Since we want to investigate which user interaction strategy supports the best user experience and usability of a mobile product recommender system, a working recommender system in the background is not necessary. Methodologies examining both recommender systems and mobile systems will be depicted. With this prototype, we will compare different user interaction strategies and find out which ones are most preferred by the users. The selected scenario is a mobile shopping recommender system. Therefore suitable clothing items have to be picked as test data to allow a realistic setting. In summary we want to give answers to the following questions:

Research Questions: Which interaction alternatives are suitable for a critique-based mobile recommender system? Which interaction strategies of a mobile shopping recommender system provide the best user experience? Do contextual changes affect the users' preferred interaction methods and how?

6.2 Designing the Test Application

The test application should allow a comparison of different user interaction design strategies for a critique-based mobile recommender system. We also want to investigate contextual changes and assess its effects on the user behavior. Each of the following subsections describes one step in the interaction design process (see *figure 6.2*), excluding the evaluation, which is particularly described in *section 6.3*. The concept, the implementation and the evaluation of the system have been published in [Lamche et al., 2015b].

6.2.1 Requirements Establishment

Based on the interaction model of critique-based recommender systems described in *subsection 6.1.1*, we can now make rough functional and data requirements to provide high usability and best user experience for each interaction activity.

1. For the task of setting initial preferences, the user can either *explicitly* specify its preferences or alternatively allow the system to automatically collect the user's preferences (e.g., based on the user's browsing and clicking behavior). In this *implicit* way, no user interaction is required.
2. In the second step, the user should be presented one or several items matching the elicited preferences, accompanied by information why this item was elected.
3. The user should be able to select an item from the presented step and either mark it as her final choice, or provide some kind of feedback on the item and/or its attribute values in order to revise her preferences and receive a new set of recommended items.

6.2.2 Designing Alternatives

Designing alternatives is the core activity of the interaction design: Actually suggesting ideas, which meet the requirements.

Setting Initial Preferences

When designing alternatives for setting the users' initial preferences regarding an item, we distinguish between two different preference elicitation techniques: Stating preferences by assigning values to several clothing item features and stating a reference product that is being searched. Design alternatives that cope with the initial elicitation of preferences were developed according to these two techniques: First, manually setting the feature values. Second, taking a picture of an item or uploading an existing one. The system should then recognize some properties of the covered item or find items similar to it based on an image search. A third technique would be an implicit preference determination. *Table 6.1* presents the alternatives with an overview of their properties regarding the acquisition process of preferences; whether all clothing attributes are clearly visible or only in a final overview; and whether it uses a mobile-specific module to perform the task of assembling preferences.

	Acquisition Process	Visibility of Item Attributes	Mobile Module Used
Take Picture	System: Image recognition	Only in overview	Camera
Upload Picture	System: Image recognition	Only in overview	Internet Access
Manually Set	Explicitly user: List of attributes	Yes	-
Answer Questions	Explicitly user: One screen per question	Only in revision	-
Implicitly	System: User-model	No	Internet Access

Table 6.1: 'Setting preferences' alternatives

Presentation

As response to the user's initial set of preferences from step one, the system has to show either *one* or *several* recommended items fitting these preferences. Additionally, if the system returns multiple items, the questions to be answered are how many items to show and how to lay them out. A recommended item, or each item in the recommendation set, must carry a specific chunk of information as to why it is recommended. A simple textual description formed from item's feature values present the explanation for designed alternatives in this system. Explaining why an item is chosen inspires user trust, according to [Tintarev and Masthoff, 2007a]. A *Comparison* presentation interface was in the design stage imagined as an additional feature to directly compare two recommended items. However, it can also serve to present the user the two best-ranked items. *Table 6.2* gives an overview of the designed and presented alternatives regarding the number of items fit-

ting on the smartphone screen, the relative image size and the level of detail of the item's description.

	Number of Items Fitting on Screen	Image Size	Visibility of Item Details
Single Item	1	Big	Complete description
List of Items	Multiple (≈ 6)	Small	Most of the description
Grid of Items	Multiple ($\approx$ 3x3)	Small	1-2 item attributes
Map	Multiple	Small	Most of the description
Comparison	2	Small	List of attribute-value pairs

Table 6.2: 'Presentation' alternatives

Giving Feedback

User feedback (also known as preference revision) is a vital component of most recommender systems, allowing a system to make better suggestions by adapting its current understanding of users' requirements [McGinty and Reilly, 2011]. In order to achieve this, several feedback strategies have been developed. Older strategies include *ratings-based* feedback, while two alternatives, *critiquing* and *preference-based* feedback, are the subject of more recent research [McGinty and Reilly, 2011]. Although the critiquing strategy is a common approach in conversational recommender systems, the idea in designing alternatives for this step is to investigate which strategy is favorable in the case of a mobile product recommender system. Therefore, the designed alternatives in *Table 6.3* incorporate the different feedback strategies and depict how they have been handled in this approach.

	Strategy	Description of Strategy
Rating Stars	Rating	Item/features get a rating between 1 (horrible) and 5 (excellent) stars.
Like/Dislike	Rating	Item/features get a 'like' or 'dislike'. Users can 'like' an item and still 'dislike' a feature of the item.
Positive/ Negative	Rating, Preference	If the item is rated positive/negative, features can also only be rated same.
Critiquing	Directional or Replacement Critiques	Directional: Attribute is in- or decreased (e.g., price); Replacement: Attribute is replaced with another value (e.g., color).
System-Alternatives	No explicit feedback	The system shows alternative items, differing in one or several feature values.

Table 6.3: 'Giving feedback' alternatives

Considering Context

One of the main goals of this chapter is to examine whether context-changes such as location, timestamp, budget, weather, companion and so on influence the user's preferences about their favorite interaction technique. Our second main goal is to conclude whether contextual changes (e.g., of location, budget or weather) imply changes on the users' preferences about their favorite interaction and interface options for a mobile recommender system. Mobile systems dispose many sensors measuring physical dimensions, but how to map this data to a concrete situation? When mapped, how to determine which contextual features are important and should be taken into account by the recommendation algorithms? We also want to find out whether a specific contextual situation can influence the user's opinion about a certain interaction technique and change her favorite way.

6.2.3 Prototypes

Interaction design involves designing interactive products. Before deploying a final version, these products have to be reviewed, graded and maybe improved, which can be achieved through prototyping.

Low-Fidelity Prototype

At this level of prototyping, the focus is on the product concept and general implementation, not on details. Low-fidelity prototypes developed for this user study are hand-sketched paper prototypes and mainly have the purpose to eliminate the least attractive design alternative in each interaction step with an online survey which will be described in *section 6.3.1*. *Figure 6.3* shows an example of a low-fidelity prototype, illustrating the *Like/Dislike* feedback alternative. We implemented each of the selected prototypes from the survey as a higher-fidelity prototype to investigate it in the final evaluation test. This way the design idea was distinguished from implementation issues that could have arisen.

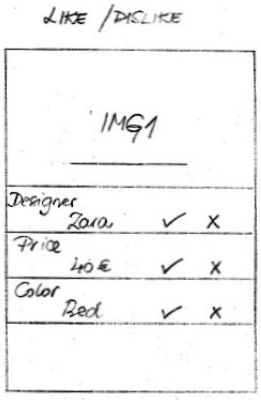
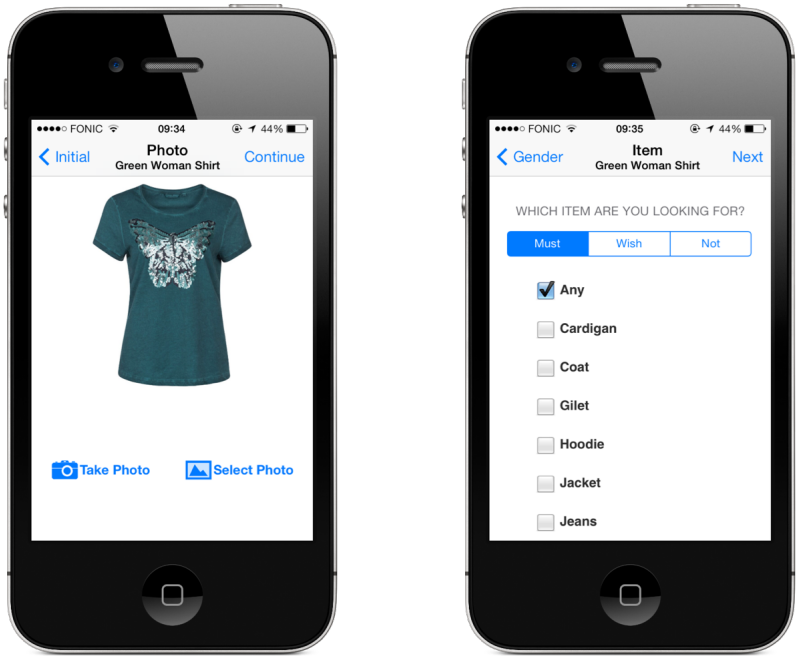


Figure 6.3: Example of a low-fidelity prototype

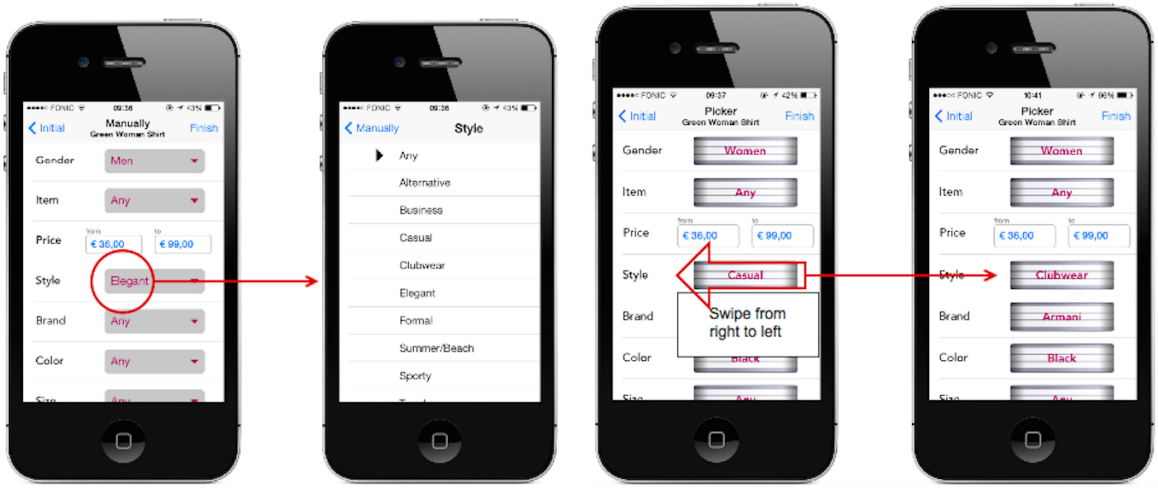
Higher-Fidelity Prototype

After designing the low-fidelity prototypes and their evaluation, a clear idea of the basic design and a fairly comprehensive list of features should be available for the development process of higher-fidelity prototypes. The prototypes are implemented as an iPhone 4S application that does not have a running recommender algorithm in the background, nor



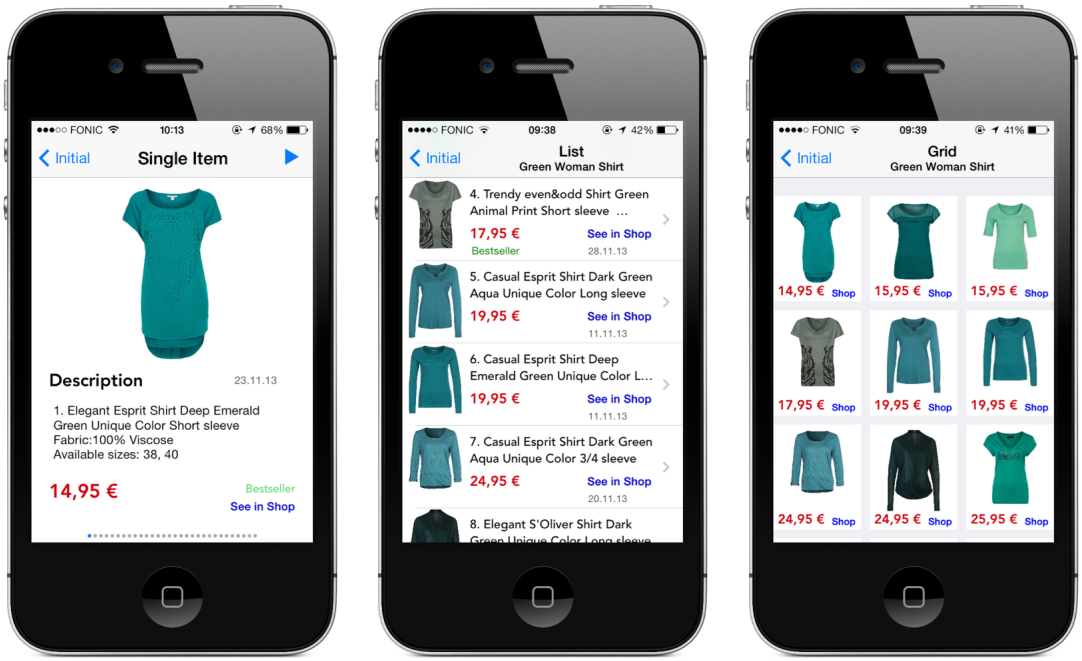
(a) "Take or Use a Picture" design (b) "Answer Questions" design

Figure 6.4: *Take or Use a Picture (a)* and *Answer Questions (b)* interaction steps



(a) "Combobox" design (b) "Picker" design

Figure 6.5: Difference between *Combobox (a)* and *Picker (b)*

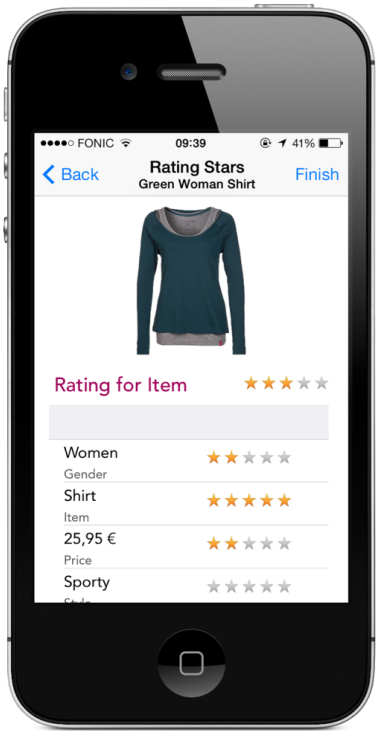


(a) "Single Item" design (b) "List" design (c) "Grid" design

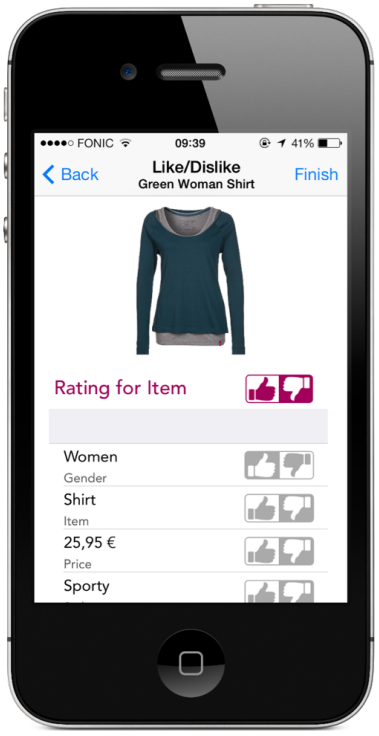
Figure 6.6: "Presentation of items" interfaces

an image recognition process. The complete interaction takes place with mocked data. We use eleven features to describe a clothing item: *Gender, Item, Price, Style, Brand, Color, Size, Fabric, Pattern, Sleeve Type* and *Length*. The *Item* value changes when alternating the gender value. The set of features is fixed, i.e., the set does not change when a different item type is chosen. Values are alphabetically ordered, but the features itself not. For a complete list of features and values see *Appendix C.2*.

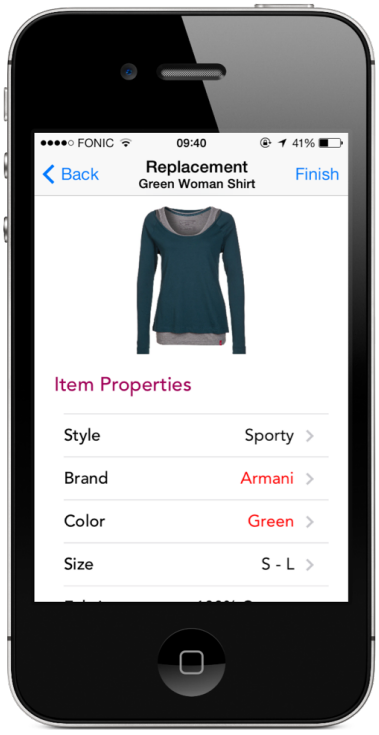
Setting preferences Four designs are implemented as higher-fidelity prototypes for the step of stating initial preferences: *Take or Use Image, Answer Questions, Manually Combobox* and *Manually Picker*. In the form of higher-fidelity prototypes, the alternatives *Take Picture* and *Upload a Picture* are merged and act as one acquisition strategy (see *figure 6.4*). That way, the user can either take a "live" image or upload an existing picture from one of the mobile phone's photo albums. The process continues with the system recognizing features from the picture and presenting them to the user in an overview list. The user can either change features that got a wrong value in the image recognition, give a value to features that were not recognized or finish the elicitation process. The *Answer Questions* strategy consists of twelve separate screens: One for each product feature and the last one as a static overview of stated values (see *figure 6.4*). The remaining two designs, *Manually - Combobox* and *Manually - Picker* are founded on the same layout idea, but differ in the domain visibility. When selecting a feature cell in the Combobox prototype, a new screen



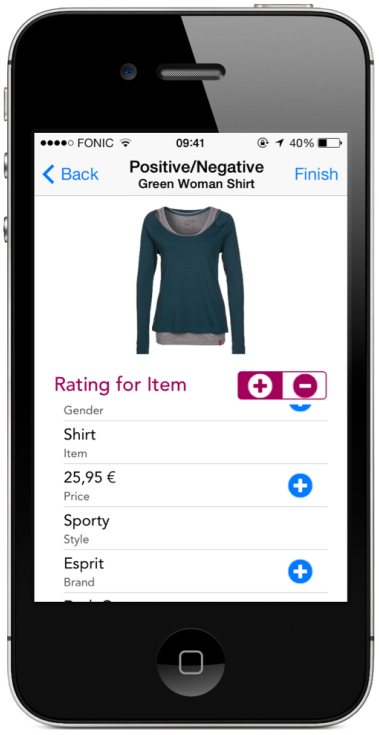
(a) "Rating Stars" design



(b) "Like/Dislike" design



(c) "Replacement" design



(d) "Positive/Negative" design

Figure 6.7: "Giving feedback" interaction steps

containing *all* feature values appears. By picking a value, the screen automatically disappears and the cell's combobox gets the chosen value. Within the Picker screen, a tap on a feature cell does nothing. A swipe action on the picker view, right-to-left, reveals a new value and hides the previous by pushing it to the invisible left. Thus the whole interaction takes place within a single window. See *figure 6.5* for a visual description.

Presentation Concerning the presentation of recommended items, the design alternatives *Single Item, List* and *Grid* were given the form of a higher-fidelity prototype (see *figure 6.6*). Each of the presentation views contains a *shop*-button, which leads the user to the shop's web page containing a more detailed description and multiple images of the item from various perspectives.

Giving Feedback The designs *Rating Stars, Like/Dislike, Positive/Negative* and *Replacement* are developed as higher-fidelity prototypes to represent the different feedback strategies. Examples for the presentation views can be seen in *figure 6.7*.

Context The context screen shows different context information (such as *Location of Shops, Currently Opened Shops, Availability of Online Shop, Budget, Season, Weather, Companion* and *Transport*) that can be included in the recommendation process when selected from the user. The user sets the values for the contextual factors *Budget, Companion* and *Transport*, the remaining factors are obtained by the system. Our contextual testing approach is derived from the work of Baltrunas et al. and aims at finding out whether users prefer different methods of interaction depending on the current contextual situation [Baltrunas et al., 2012]. *Figure 6.8* shows an example of the context user interface.

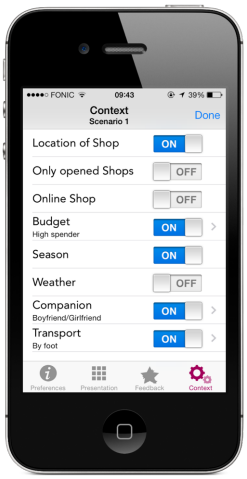


Figure 6.8: "Context Setting" design

6.3 User Study

The main two goals of the evaluation are to find out which interaction and interface possibilities provide the best usability and user experience in supporting the interaction model steps (see *section 6.1.1*) and to conclude whether contextual change implies changes on the users' preferences about their favorite interaction and interface options for a mobile recommender system. Despite the fact that we only evaluate the interaction design, without providing a working recommender system in the background, its aim still targets the usability and user experience domain of mobile recommender systems. Because of the wide option space defined in (*section 6.2.3*), the user study could take a long completion time. In order to reduce subject fatigue, one extra step between the interaction process and user-study was introduced. Goodman et al. depict a *value* survey, investigating what people

find important and is usually run before major further moves [Goodman et al., 2012]. We therefore first carry out an *online survey* to narrow down the design alternatives and reduce subject fatigue in the following user study. Based on these results, we conduct the *user study*. A user study requires the recruitment of test subjects who perform a specific task while observing their behavior and collecting a number of quantitative measures. Common measures are the users' effectiveness and efficiency while performing the task. A one-on-one usability study can quickly provide a great amount of information on how a product will be used.

6.3.1 Online Survey

The online survey served as a preliminary study and had the simple task of asking people what their preferences are regarding interfaces and which interaction functionalities they would value as important in a mobile shopping recommender system. The goal of the survey was to collect fast user opinions on the developed paper prototypes, which were presented as sketches to the user (see *subsection 6.2.3*). The survey questionnaire disposed a total of 38 questions, divided into four blocks. Besides demographic questions, users were asked about preferences for eliciting item preferences, presenting recommended items and critiquing/giving feedback on an item. The survey was online for three weeks. In total 46 people participated, 27 males and 19 females. The participants were 26.7 years old in average, ranging from 20 to 57. Based on the results of the online survey we included the most favorite interfaces and functionalities regarding a mobile shopping recommender system in the higher-fidelity prototype. Concerning the importance of additional features, users showed (among other things) preferences for a "see more recommendations" button, a possibility to modify the initial preferences, a keywords search field and explanations of recommendations. The complete questionnaire is available in *Appendix C.1*.

6.3.2 The Testing Procedure

Based on the results of the online survey, the developed higher-fidelity prototype described in *subsection 6.2.3* was judged within a user study with respect to the usability and user experience. The study consisted of collecting data from three categories. First, the study data was logged by the higher-fidelity application during its execution. Thus, we were able to collect interaction data (e.g., the time users needed to finish the given task with an interface or the currently set preferences) in order to analyze and understand the user behavior [Rogers et al., 2011]. Second, during the entire time of the user study, the examiner took notes on the users' comments (*Think Aloud method*). Third, a usability questionnaire collected general data about the person and the imprinting on the interfaces. During this study, the user was set up with three distinct tasks, one for each step in the product recommender interaction model. The application was made to act as a mobile clothing shop, a commonly known domain. Each developed design ran twice: First for a *green women's T-shirt* and afterwards for *beige men's trousers*. These items are presented in *figure 6.9*. The questionnaires examined the differences in the subjective satisfaction

Figure 6.9: Items that were looked for in the user-study

with the user-friendliness of the designs. Another questionnaire collected demographic data and the level of shopping experience of individual trial participants (see *Appendix C.4*). As the underlying base for the development of the questionnaire, the framework *ResQue* was used. ResQue consists of 13 constructs and a total of 60 questions, divided into four main dimensions [Pu et al., 2011a]. The following perceived qualities of recommender systems were investigated within our questionnaire: *Ease of Use*, *Interaction Adequacy*, *Interface Adequacy*, *Control*, *Attitude/Overall Satisfaction*. With all questions not being suited for each developed design, the setup was slightly different for distinct interaction activities. Participants stated their opinion with a 7-point Likert scale (from 1, strongly disagree to 7, strongly agree). The complete form of all questionnaires with their belonging questions can be found in *Appendix C.3*.

The developed designs were evaluated in a lab-based study, which lasted for about an hour. As a recommender system has a wide audience, no specific user group was targeted. Each of the total eleven interfaces that were examined had an accompanying video on the interaction possibilities, shown before the start of every interface design. After the task completion for both items, users were asked to fill out a questionnaire about the usability of the design in relation to ease of use, interaction and overall satisfaction. When all tasks associated with one interaction step were performed, the user was asked to choose her favorite and least favorite interface design from that group. The order in which the interfaces were presented to the subjects was randomly allocated, however staying within the interaction step. By changing the order, learning effects were avoided and each interface could be objectively evaluated.

The first group of interfaces served the user to explicitly describe the item having in mind to the system with a feature-value list consisting of eleven semantic features. The second interface group included interfaces presenting an initial set of recommended items. A set of 30 items for each initial item was mocked, acting as the recommended items. The set was sorted by the items' prices because participants expressed this feature as being the most important one in the previously conducted online survey. The user's task was to lookup for the item most likely similar to the item described in step one. For the final

interaction step, a random item was shown to the user that had to be compared to the one looking for by rating preferences or replacing attribute values. In order to examine context influence on the process of decision-making, the participant was asked to imagine herself in two certain context situations described in detail in *section 6.3.3* (as in [Baltrunas et al., 2012]) and to determine which context factors are of high importance to her in that specific case and activate those factors within the context screen. The participant was asked once more to choose her favorite designs from each interaction step, but now according to the imagined context.

6.3.3 Results

The random sample included 21 evaluators, aged between 19 and 39 years with an average age of 26.5 years. The gender distribution was rather balanced with 52.4% of users being male and 47.6% being female. The means of the measured values, as well as the standard deviation (stdev) of the examined interfaces are shown in the corresponding tables. An analysis of variance (ANOVA) at a significance level of 0.05 was performed to calculate the p-value. The p-value defines how significant the results are and is shown in the last column of the table. Significant values are printed bold. Due to space reasons, this chapter only includes diagrams illustrating the overall users' satisfaction.

Setting preferences

The first task included describing an item to the system. *Figures 6.4* and *6.5* show the corresponding interface designs. Within this category, all results are statistically significant. By tracking the time from the beginning that occurs with the *green shirt* item to the end of describing the *beige trousers* item, the task could have been completed in around 2 minutes for the three interfaces *Take Or Use Picture*, *Combobox* and *Picker*, while *Answer Questions* needed almost 6 minutes (see *table 6.4* for exact values).

When asked about the ease of interaction to describe an item to the system (on a 7-point Likert scale, 7 the best, 1 the worst rating), the participants' average rating was very similar among the three systems *Answer Questions, Combobox* and *Take or Use Picture*. Participants expressed much less satisfaction for the *Picker* alternative.

When asked whether the design presents an adequate way to express preferences, almost all ratings for *Answer Questions, Combobox* and *Take or Use Image* were above four. Picker was the only alternative rated less than four on average. These results are almost mapped to the issue of whether the design offers an adequate way to summarize preferences. In this context, some participants stated that they wished a dynamic overview design, i.e., to be able to jump to a certain question when a preference is selected in the overview.

Looking at the rating distributions on the level of control participants perceived when telling the system what an item they want, as well as whether they can use the interface

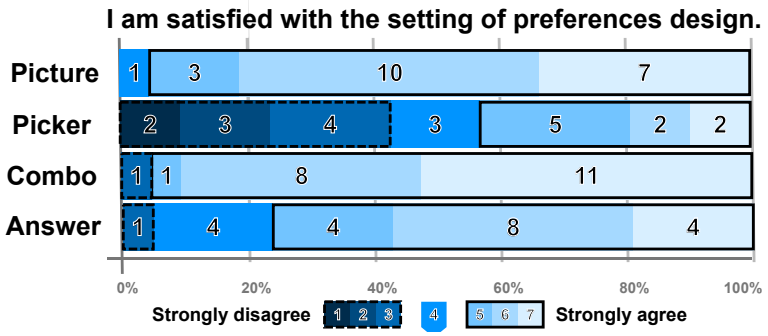


Figure 6.10: Overall satisfaction with the setting of initial preferences designs

for a long time without any input errors, no interface has fallen into ratings less than four in average. *Combobox* provided the highest feeling of control, *Picker* the lowest.

When asked what the overall impression of the system was, participants expressed high sympathies for *Combobox* and *Take or Use Picture*, while *Picker* was rated worst (see *figure 6.10*). Concluding from the participants' comments besides being complicated to interact with, *Picker* did not provide sufficient visibility of the item attributes.

Result 1: 'Combobox' is the favorite preference elicitation strategy and 'Picker' the worst concerning ease of use, adequacy, control, accuracy and satisfaction. 'Take/Use Picture' is always second place and 'Answer Questions' third, with the exception of the accuracy category. It is worth mentioning that 'Answer Questions' needs almost the triple of time compared to the other preference elicitation strategies.

	Answer Questions	Combobox	Picker	Take/Use Image	p-value
Time	5.87 min stdev = 2.58	2 min stdev = 1.07	2.47 min stdev = 0.82	1.87 min stdev = 0.73	**<.001**
Ease of Use	6.14 stdev = 1.15	6.57 stdev = 0.75	4.38 stdev = 1.8	6.29 stdev = 0.78	**<.001**
Adequacy	5.43 stdev = 1.33	6.24 stdev = 1.04	3.9 stdev = 1.97	6.1 stdev = 0.77	**<.001**
Control	5.52 stdev 1.36	6.38 stdev = 0.80	5.19 stdev = 1.21	6.09 stdev = 1.09	**.004**
Error-Free	5.57 stdev = 1.57	6.29 stdev = 1.06	4.67 stdev = 1.85	4.76 stdev = 1.61	**.006**
Satisfaction	5.48 stdev = 1.17	6.33 stdev = 1.0	3.95 stdev = 1.8	6.1 stdev = 0.83	**<.001**

Table 6.4: A comparison of the user study's results concerning the preference elicitation

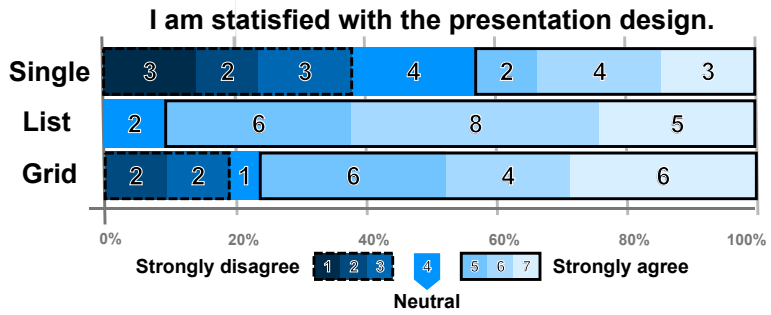


Figure 6.11: Overall satisfaction with the presentation designs

Presentation

We evaluated three different presentational interface designs (see *figure 6.6*). By looking at the average time of choosing the best suited item for each design alternative, there is only a subtle difference with participants completing their session (around 1 minute). *Grid* wins in this category, *List* is second and *Single Item* third place. *Table 6.5* shows the exact time values, having a statistically significant difference.

When asked if it was easy to use the interface, *List* performed best on average, followed by *Grid* (second place) and *Single Item*. This rank order also reflects the perceived overall satisfaction, which is considered as significant (see *figure 6.11*). Here, the span of ratings for *Single Item* was larger than in *List* and *Grid*, including very high values, but also very low values and reaches last place.

Things change when looking which design alternative has the most adequate interface as well as if the interface provides sufficient information. *Grid* drops down to last place when asked if it provides sufficient information and is second when asked about the layout's adequacy. *List* is rated best in both categories, while *Single Item* is considered as least adequate but second concerning the information content. A statistically significant difference was found between the perceived adequacy of the interfaces.

Participants scrolled more often in order to find and select the most appropriate item when using the *List* interface compared to the *Grid* interface.

Result 2: Users favored the presentation of items in a 'List' view, while the 'Grid' view was ranked second and the 'Single Item' view worst in relation to ease of use, adequacy and satisfaction. However, 'Grid' is regarded as not giving sufficient information compared to the other two designs.

	Grid	List	Single Item	p-value
Time	0.80 min stdev = 0.89	1.05 min stdev = 0.59	1.24 min stdev = 0.66	**<.001**
Ease of Use	5.95 stdev = 1.53	6.43 stdev = 1.12	5.52 stdev = 1.78	.215
Adequacy	5.33 stdev = 1.65	6.05 stdev = 0.92	4.86 stdev = 1.9	**.037**
Sufficient Info	4.71 stdev = 1.76	5.86 stdev = 1.35	5.52 stdev = 1.25	.066
Scroll-Downs	10.89 stdev = 5.63	15.61 stdev = 12.65	- -	.212
Satisfaction	5.24 stdev = 1.64	5.76 stdev = 0.94	4.14 stdev = 2.03	**.008**

Table 6.5: A comparison of the user study's results concerning the different presentation interfaces

Giving Feedback

We implemented four different feedback strategies within our higher-fidelity prototype (see *figure 6.7*). Regarding the time measurement, the *Positive/Negative* way of feedback scored best followed by *Replacement, Rating Stars* and *Like/ Dislike* (which is the most time consuming feedback strategy). However, the difference is almost imperceptible (see *table 6.6* for exact values of the time measurement).

Participants were asked to evaluate the ease to use the interface. The *Positive/Negative* and *Like/Dislike* designs had to be described to almost all users after showing the interaction video. *Positive/Negative* needed extra explanations for its logic, while the *Like/Dislike* interface coloration was not easy to distinguish from. This reflects in the ratings participants have given (see *table 6.6*). Participants complained mostly on the restriction to positive- or negative-only ratings, while the main plague of the *Like/Dislike* interface was its lack of a default, neutral rating option, as well as its color scheme that was misguiding. On the other hand, *Rating Stars* got the complaint of the stars being too small.

Continuing, participants were asked to consider the interaction adequacy of revising preferences with each feedback strategy. The *Replacement* design achieved first place, *Positive/Negative* was elected worst. In between are *Like/Dislike* and *Rating Stars*. The overall satisfaction maps the interaction adequacy almost one-to-one (see *figure 6.12*). The differences of the calculated satisfaction as well as of the interaction adequacy are statistically significant.

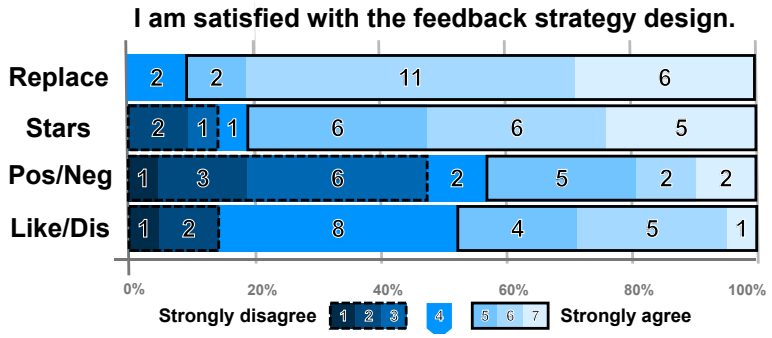


Figure 6.12: Overall satisfaction with the feedback strategy designs

Result 3: Regarding the ratings for ease of use, adequacy and satisfaction, the 'Replacement' critiquing strategy is ranked best, followed by 'Rating Stars', 'Like/Dislike' and 'Positive/Negative', being the least favorite strategy. However, 'Positive/Negative' is the most efficient critiquing approach in terms of time.

	Like/ Dislike	Positive/ Negative	Rating Stars	Replacement	p-value
Time	1.33 min stdev= 0.55	1.12 min stdev = 0.76	1.31 min stdev = 0.58	1.27 min stdev = 0.36	.788
Ease of Use	5.71 stdev = 1.59	5.67 stdev = 1.68	6.19 stdev = 1.12	6.38 stdev = 0.80	.237
Adequacy	4.38 stdev = 1.99	3.71 stdev = 2.05	5.14 stdev = 5.14	6.05 stdev = 1.28	**.002**
Satisfaction	4.48 stdev = 1.5	4 stdev 1.7	5.33 stdev = 1.53	6 stdev = 0.89	**<.001**

Table 6.6: A comparison of the user study's results concerning the feedback method

Context

The participants were also asked to imagine themselves in two distinct context situations, denote the important contextual factors and pick a favorite from the evaluated variants, but now according to two different situations. The two scenarios were described as follows:

Scenario 1: You have an important meeting in 30 minutes, but you just spilled coffee all over your shirt. You are in panic looking to buy a new one. You don't care about money, you just need a new white shirt as fast as possible. While walking around the neighborhood to find a shop, you are using the recommender app to find you a perfect match nearby.

Scenario 2: You are at home, surfing the space of internet to buy your mom a present for Christmas which is in two weeks. You are looking for a nice white woolen sweater, winter-appropriate. Your budget is unfortunately very limited.

While under pressure and having a reference product, almost all participants would either *Take an Image* of an item or describe it with the *Combobox* design. However, without the pressure factor and with the lack of a reference product (scenario 2), the number of favorite votes for the *Take or Use Image* design drops down to zero. On the other hand, Combobox increases its advantage, with the *Answer Questions* design following. This means that, when asked about favorites according to context, 81% parted with a 48:33 ratio between participants that changed their favorite vote for one context scenario and for both.

When presented a resulting set of items dependent on context, the ratio in favor of *List* and *Grid* changes. For the first situation, *Grid* has slightly more votes (11 vs. 8), while in the second situation; *List* has 10 votes and 3 more than *Grid.* In the terms of change, 19% of participants did not change anything, 43% changed at least one, while 38% changed both of their favorites.

Not much changed regarding the overall rating of the favorite feedback strategy. *Replacement* is the participants' favorite revision option. However, only a quarter of participants stood up to their previously rated favorite design: 48% changed in both context scenarios, and 29% in one of them.

Result 4: Only when under pressure, 'Take or Use a Picture' is a very popular preference elicitation strategy among smartphone users. Also the participants' favorite presentation interface as well as favorite method to provide feedback depends on the context situations.

6.4 Conclusion and Next Steps

This chapter described an accomplished interaction design process regarding mobile product recommender systems. The process resulted in developing eleven interaction design alternatives on an iPhone, categorized into three interaction activities: The initial preference elicitation process, the presentation of the resulting recommendations set and the

preference feedback process. As a result of an executed user study evaluating the implemented interactive designs, we could inter alia show that the *Combobox* preference elicitation strategy, the *List* view, as well as the *Replacement* feedback strategy are suitable for a critique-based mobile product recommender system concerning ease of use, adequacy and overall satisfaction. The study also showed that contextual change heavily influences the participants' choice of favorites. *Figure 6.13* presents the way in which the evaluated design alternatives could be implemented by mobile recommender system developers. Our developed mobile recommender system now consists of a critique-based recommendation algorithm, a stereotype user model and evaluated user interaction methods that support the user experience. Now that we have proven that contextual change determines the process of decision-making and therefore should be considered by mobile recommender systems, the assessment of context-relevance, as well as the integration of context-awareness into a mobile recommender system require further studies. Within the following chapter we will therefore come up with an approach how to acquire context-relevance as well present a strategy that allows the consideration of context in a mobile recommender system.

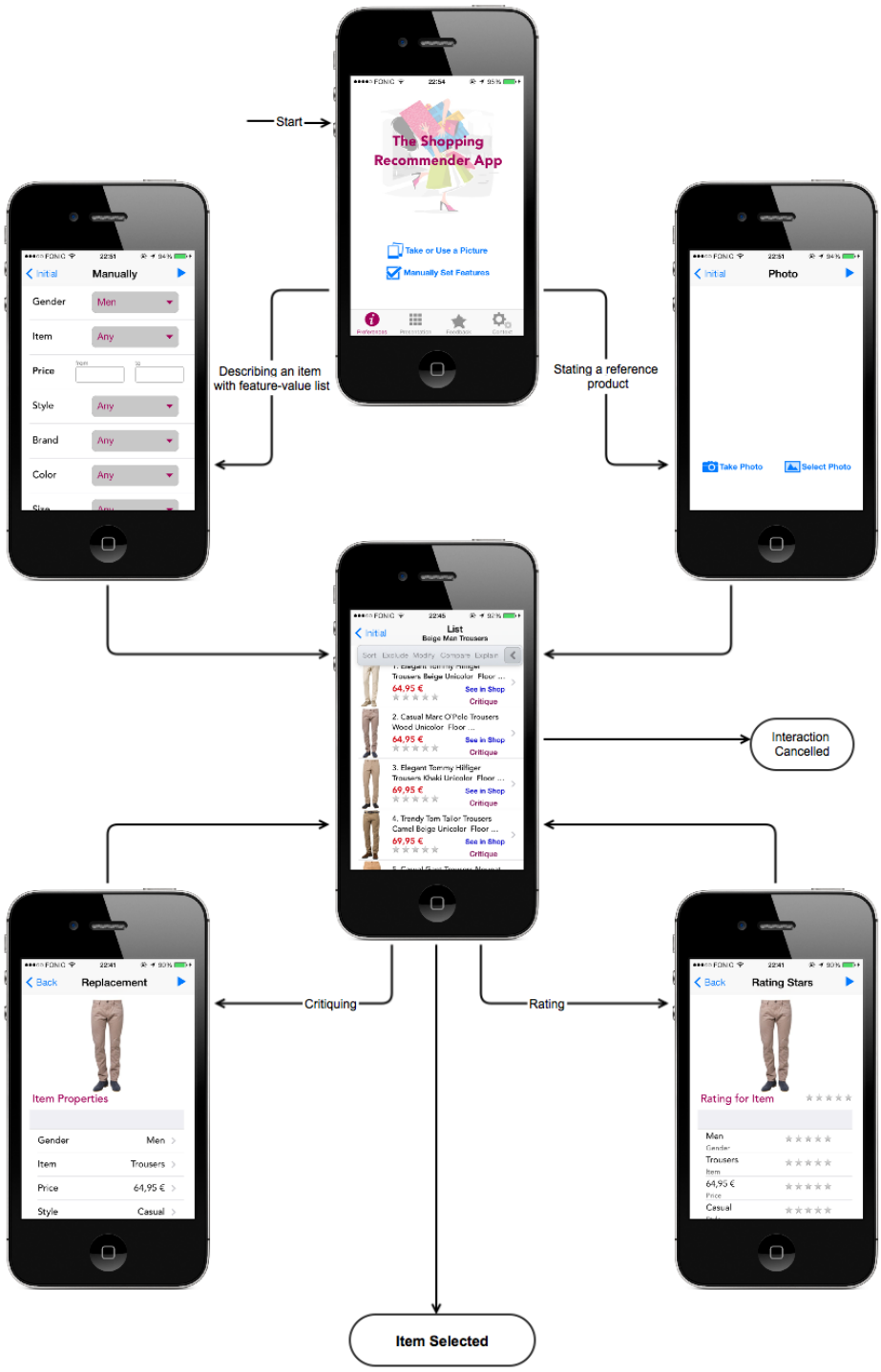


Figure 6.13: Interaction design of a mobile shopping recommender system implementing the derived results

7

Context-Awareness

This chapter explores if the integration of context-aware information can improve mobile recommendations. We first assess the influence of different context factors on the shopping behavior by conducting a preliminary study. Based on these results, we develop and evaluate a context-aware mobile shopping recommender system to find out if contextual information such as weather, budget and shopping intent can predict the user's current shopping interest to improve accuracy, efficiency as well as the user experience of a mobile shopping recommender system. The chapter is divided into four sections. The current one (1), gives a general introduction to context-aware recommender systems and summarizes related work. The second section (2) introduces our methodology for building a context-aware mobile recommender system. In a first experiment, we acquire the context relevance. We then integrate the contextual information into an existing mobile shopping recommender system based on Active Learning. We present the user study and discuss its implications in section (3). The final section (4) concludes and hints towards future research potential.

7.1 Motivation

With the improvement of the GPS technology the location of smartphones can be measured within 10 meters and is tracked periodically. Mobile recommender systems can use information as well as other sensor data to deliver accurate suggestions according to the user's location, e.g., if the user is close to a point of interest. The results of the survey we conducted in *chapter 6.* showed that the inclusion of mobile context may lead to more accurate recommendations in our proposed approach. Context-aware recommender systems (CARS) are systems utilizing the user's context such as the user's position, weather or social environment to deliver accurate suggestions. This is especially desirable in an exploratory scenario where the user does not exactly know what she is looking for (e.g., in a shopping or tourism scenario). A context-aware recommender system could for example

recommend the "Deutsches Museum" rather than a long walk along the Isar if a tourist spends a rainy day in Munich. Few studies have investigated the integration of contextual information into an Active Learning mobile recommender system. The goal of this chapter is therefore to develop and evaluate a context-based mobile shopping recommender system to find out if context-aware information such as weather, budget and shopping intent can predict the user's current shopping interest to improve accuracy and efficiency of a mobile shopping recommender system. We first evaluate which kind of context information is relevant in a mobile shopping recommender system and how this information can be utilized to improve recommendations of clothing items in a context-aware recommender system. By integrating contextual mobile information into the recommendations we expect that the recommended items are better and therefore customers are more satisfied with the recommender system.

7.1.1 Context-Aware Recommender Systems

The consideration of the current mobile context can improve the accuracy of a recommender system. To understand the concept and aim of context-aware recommender systems we will discuss the term context and how it can be integrated into a CARS, as well as give an overview of existing approaches in the literature.

Context

A widely used definition in the area of context-aware applications is the definition by Dey:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [Dey, 2001, p. 5].

Dey defines context as relevant information for an interaction between a user and an application. Therefore, if the context of an entity shall be defined, it is necessary to ask which information is relevant to the situation. Adomavicius et al. divide context into four sub-categories [Adomavicius and Tuzhilin, 2011]:

Physical context can be described as the state of the environment. It includes time and position as well as weather or light [Adomavicius and Tuzhilin, 2011]. Current smartphones include many sensors to read the physical context. These sensors include an ambient light sensor (adapts the brightness of the display to the environment), a proximity sensor (detects when the smartphone is close to the ear, to turn off the display), a GPS sensor, an accelerometer, a compass, a gyroscope (measures or maintains the orientation), a back-illuminated sensor (a technique originally intended to improve pictures taken by the camera) and a microphone. All these sensors can

be used to digitize the features in the user's surroundings like incoming light, noise or magnetic field [Lee and Kwon, 2013].
The physical context also contains information about the weather [Adomavicius and Tuzhilin, 2011]. This information normally is not directly measured, but extracted from other sources such as weather information websites.

Social context can be defined as "the presence and role of other people around the user, and whether the user is alone or in a group when using the application" [Adomavicius and Tuzhilin, 2011, p. 74]. This can also involve "the social network of the user, buddy lists, past interactions etc." [Woerndl and Groh, 2007, p. 123].

Modal context describes "our presence of mind, when we perform a task" [Fling, 2009, p. 54]. The modal context is the driver of the user's actions and how she interacts with the system in order to accomplish her goals [Fling, 2009]. This may also include the user's mood and experience. It is often necessary to derive conclusions about the user's modal context from her actions [Adomavicius and Tuzhilin, 2011]. For example by allowing the user to criticize items in a critique-based recommender system, it can be assumed that she gets closer to the fulfillment of her task or goal.

Interaction media describes the device (and its properties) with which the user interacts [Fling, 2009]. It also includes the physical restrictions of the device (like processor speed or display size). For example a smartphone with a larger screen can be better used to display videos, as it is more convenient to view them, than on ones with smaller screens. Nevertheless, media context is not only about the device and its properties, but also about which applications are installed on the device [Pushpa and Venkataram, 2011]. Also the type of media that is browsed or personalized is part of the interaction media context. Among these can be music, text, images or queries made to the recommender system [Adomavicius and Tuzhilin, 2011].

Although there are many other context classifications such as by Chen and Kotz [Chen and Kotz, 2000], Jung [Jung, 2009] or Kofod-Petersen and Aamodt [Kofod-Petersen and Aamodt, 2003], we focus on the classification by Adomavicius et al. [Adomavicius and Tuzhilin, 2011], as according to them, they are the most commonly exploited in CARS and are therefore assumed to provide the most benefit.

Context Elicitation

One problem for context-aware recommender systems is how to detect which context factors are important for the specific recommender systems. For the detection of context factors a domain expert or existing research in this area could be used to get relevant context factors. Adomavicius and Tuzhilin argue that it is also possible to obtain the context relevance automatically via statistical or machine learning methods, if there is a set of ratings with context data [Adomavicius and Tuzhilin, 2005]. However, this implies that a large number of context factors was measured in advance and is available. Baltrunas et al. propose that a large set of possible context factors is selected by an expert. This set should

contain all possible context factors and conditions. The set is then used in an application where users can state whether a context condition does influence the rating of the item positively, negatively or not at all. By executing this test the users give the recommendation system designers valuable insights on which context factors and conditions could be relevant and how relevant they are for giving recommendations. This approach has the advantage that no existing dataset is necessary. However, it might have the disadvantage that test participants are not really aware of what influences them in their decisions. Hence, the results from surveys like this could be wrong [Baltrunas et al., 2012].

Another problem for context-aware recommender systems is how to retrieve enough context-based ratings in different context scenarios. Baltrunas et al. propose designing an application that encourages the users to imagine certain context conditions. They investigated the relationship between contextual factors and item ratings in a tourist scenario. The authors developed a web tool for acquiring subjective ratings regarding points of interest in a mobile scenario within a specific context. Users were asked if a specific context factor (e.g., winter season) has a positive or negative influence on the rating of a particular item. Second, users were asked to rate example contexts and recommendations. The more influence-able a context factor seemed to be (according to the results of the first step), the more contextual conditions specifying this factor were generated. These imagined ratings could be used as initial ratings in the database, such that the cold start problem is minimized. Based on these results, a predictive model that can be trained offline, was developed. Results show that influencing context factors for points of interests are inter alia *distance, season, weather, time, mood and companion* [Baltrunas et al., 2012]. This methodology seems to be a very promising approach to acquire contextual ratings, however ratings were only acquired for a travel planning recommender system and the generated ratings of this work can't be directly applied to a mobile shopping scenario.

Once the relevant context factors and possible conditions are identified, the system designer must decide how to elicit the context conditions. There are three different ways in which this is possible [Adomavicius and Tuzhilin, 2011]:

Explicit elicitation of context describes directly asking the user what the context looks like. For example a system could ask the user whether it was raining, or who is accompanying her.

Implicit context elicitation is about measuring the context without asking the user. The context data is then measured via sensors, e.g., the microphone, the GPS sensor or others.

Inferring context describes a method by which the user behavior is analyzed and certain context conditions are derived from this behavior. Often methods like statistical analysis or data mining are used to gather these information.

Ideally the application is able to measure all context data implicitly. However, sometimes the user has to be asked what her context looks like. Inferring this data might be error prone and could lead to wrong conclusions.

Integrating Context into Recommender Systems

Context-aware recommender systems (CARS) integrate context into the recommendation process. This process can be described by this three dimensional recommendation function [Adomavicius and Tuzhilin, 2011]:

$$R\colon User \times Item \times Context \rightarrow Rating \tag{7.1}$$

The rating function (R) considers the *Context* (which is defined by all the different *Context Factors*) and recommends items of the item set (*Item*) to a user by predicting the rating that this user would give to an item. Context complicates the recommendation process as items can be rated in different contexts. An umbrella for example can be rated at good weather conditions very highly, due to the fact that it looks nice or is small. However, if it was raining the same umbrella could get a bad rating, due to the fact that it breaks at the slightest wind. So the context in the rating function brings additional complexity as the recommendation algorithm does not only have to match users with items, but also with the context. Adomavicius and Tuzhilin identified three different points in the recommendation process where context might be incorporated into the process:

Contextual Modeling: The recommendation algorithm is altered such that it includes the context and already considers it when calculating recommendations.

Contextual Pre-Filtering: The current context is used to select only the most relevant data from the dataset.

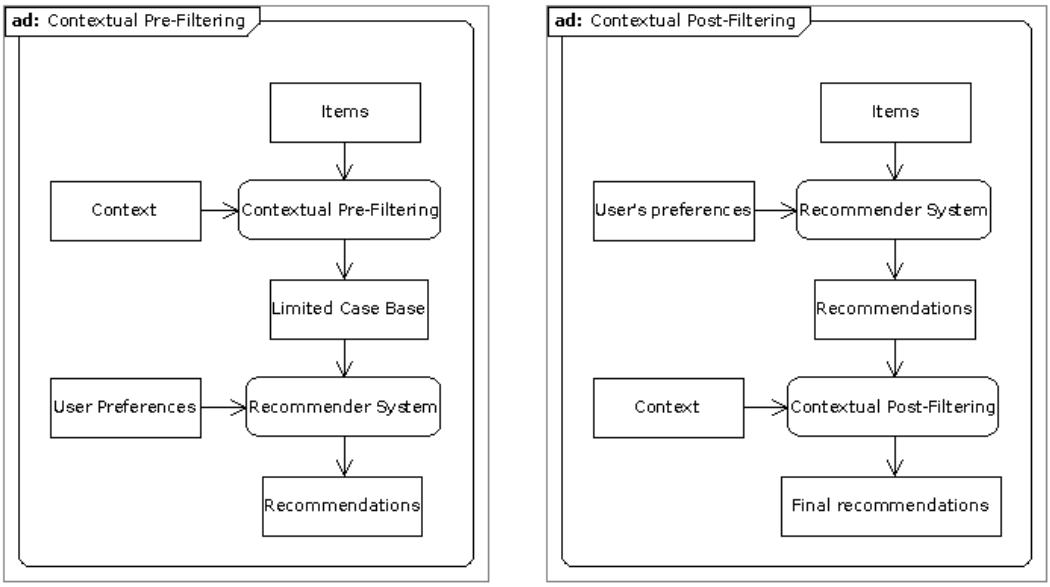
Contextual Post-Filtering: The context information is ignored during the recommendation process, only the resulting set is contextualized.

The difference between contextual pre-filtering and contextual post-filtering is illustrated in *figure 7.1*. All of these approaches have their specific strengths and weaknesses. However, it is also possible to combine multiple context-based algorithms [Adomavicius and Tuzhilin, 2011].

7.1.2 Existing Approaches

As described in the previous subsection, the standard recommendation algorithms need to be adapted when context should be integrated. This section describes how different authors reason about context from an algorithmic perspective.

Panniello and Gorgoglione compared several different contextual modeling approaches with each other, the un-contextual recommender and some post-filtering methods (see *subsection 7.1.1*). They found that contextual modeling might outperform post-filtering methods based on mean absolute error and F-measure. The contextual modeling algorithms they used do not distinguish themselves much from each other. The only difference



(a) Contextual pre-filtering (b) Contextual post-filtering

Figure 7.1: Comparison of contextual filtering processes

in the contextual-modeling approach was that the algorithms selected the nearest neighbors in different ways. As they argue the results are expected, as the nearest neighbor algorithm does not rely that much on the selection criteria for neighbors [Panniello and Gorgoglione, 2011]. In a subsequent study by Panniello et al. they compared all their previous algorithms. They compare the exact pre-filtering, the weight and post-filtering methods and their contextual modeling methods against each other according to accuracy and diversity. They found out that none of the recommendation algorithms is superior in all datasets, though one of their contextual modeling alternatives, which does not restrict the neighbors to select, performed best [Panniello et al., 2014]. In general the selection of an algorithm largely depends on the dataset. Some algorithms present more diverse items, whereas others are more accurate. For the purpose of this thesis an algorithm showing more diverse items is preferable as we expect that the process of searching clothes is explorative.

Baltrunas and Ricci propose a technique they call item splitting. In this method they annotate the ratings with the context information and determine significant differences between the contexts. If there are any, the item is virtually split into two (or more) items, from which then only the most fitting item is given to the recommender. This means that though an item such as a trouser physically only exists once, it logically exists several times (with different ratings) to ease the creation of context-aware recommendations [Baltrunas and Ricci, 2009].

Codina et al. determined semantic relationships between context conditions and enhanced a content-based recommender system with this information [Codina et al., 2013].

Pushpa and Venkataram developed a model they term C-IOB (Context-Information Observation Belief) to analyze the context data. This model is inspired by cognition science. The gathered context information is formulated into observations (describing what the user is doing, or what states of the environment are important). The generated observation is then used to derive a belief about the user or the context, which helps them to target their recommendations [Pushpa and Venkataram, 2011].

Yap et al. use Bayesian networks to learn a minimal model about the user's context preferences and derive recommendations from this [Yap et al., 2007]. With their Bayesian networks they want to capture dependencies between context factors and cope with missing or noisy context inputs. Ciaramella et al. do not look at the current context, but the context history and use a genetic algorithm to better learn the user's preferences [Ciaramella et al., 2010]. Dao et al. also use a genetic algorithm, but only to detect the nearest neighbors for a collaborative filtering approach [Dao et al., 2012].

According to Bettini et al., it is also common to reason about context to gain further information on the context. For this task different techniques for reasoning on uncertainty such as fuzzy logic, probabilistic logic, Bayesian networks, Hidden Markov models or Dempster-Shafer theory can be used [Bettini et al., 2010].

Considering all different algorithmic approaches for contextual filtering, we decide to use a nearest neighbor algorithm for our context-aware mobile recommender system. Nearest neighbor algorithms are easy to understand and flexible in their usage. Furthermore, they can be easily adapted to newly introduced parameters. Nearest neighbor algorithms mainly differ in the used distance metric. A distance metric is used to determine how close two data points (items, users, contexts, etc.) are. The distance metric used in this thesis has to be able to cope with all kinds of data, such as nominal, ordinal, interval or ratio scale data. Hence, we will analyze possible distance metrics for nearest neighbor algorithms in more detail. So far, no research exists that analyzed all the contextual factors that might be useful when recommending clothing items from different stores for mobile shoppers and integrated these factors into the recommendation algorithm. A mobile application using such an approach could help the user detecting new (formerly unknown) brands or stores and find clothes matching the user's fashion style. Compared to existing mobile recommender systems, clothing items are different in the way, that they frequently change. Such a recommender system has to be frequently trained or being able to provide good recommendations on a sparse dataset. We therefore first acquire the relevant context factors in a mobile shopping scenario and then come up with a promising approach how to integrate this context into the recommendation process.

7.1.3 Goals

In order to integrate context-awareness into the existing recommender system, it first has to be defined what exactly "context" means for a mobile shopping recommender system. Besides that, we have to evaluate which context factors should be used to model the context and how these factors can be implemented into the system. The implemented context-

awareness can then be tested in the user survey. Biases in favor of the context-aware recommender system shall be avoided by comparing the CARS with a baseline system. By developing a prototype and executing a user survey, the following research question shall be answered:

Research Question 1: How can context-awareness be integrated into an existing mobile shopping recommender system which uses an Active Learning strategy?

In order to evaluate whether context-awareness is necessary and if users benefit from it the following research question shall be answered in addition:

Research Question 2: How is the context-aware mobile shopping recommender system perceived by users compared to a recommender system that does not consider context?

7.2 Designing the Test Application

The prototype should allow a comparison of the baseline to a context-aware recommender system in order to find out how the integration of mobile context is perceived by the users and if it generates more accurate suggestions. The context-aware approach, the implementation as well as the evaluation of the test application have been published in [Woerndl and Lamche, 2015].

7.2.1 Acquiring and Integrating Context Relevance

Adapting the recommendations to the user's current contextual situation requires an understanding of the relationship between user preferences and contextual conditions. A preliminary study needs to be designed which investigates how the influence of each contextual factors changes the user's purchasing decision for clothing items and thus provides quantitative measurements that can be used as weighted attributes in the similarity measurement in the recommendation algorithm. A promising approach is presented in [Baltrunas et al., 2011] and is therefore adopted to assess the context relevance. We first selected an initial set of contextual factors and conditions (values of the factors) referring to existing literature about context-aware applications. The main context factors are: Distance, day of the week, temperature, time available, transport, weather, time of the day, crowdedness, intent of purchase, companion, season and budget. We then retrieved almost 4000 middle-priced clothing items from a popular online shop, which were randomly assigned to a list of offline stores available in Germany. Next, a simple web application was developed for acquiring the relevance of the selected contextual factors for different clothing categories. This web application randomly presents a clothing category and asks the user to imagine herself being under a randomly chosen contextual situation and select the respective influence

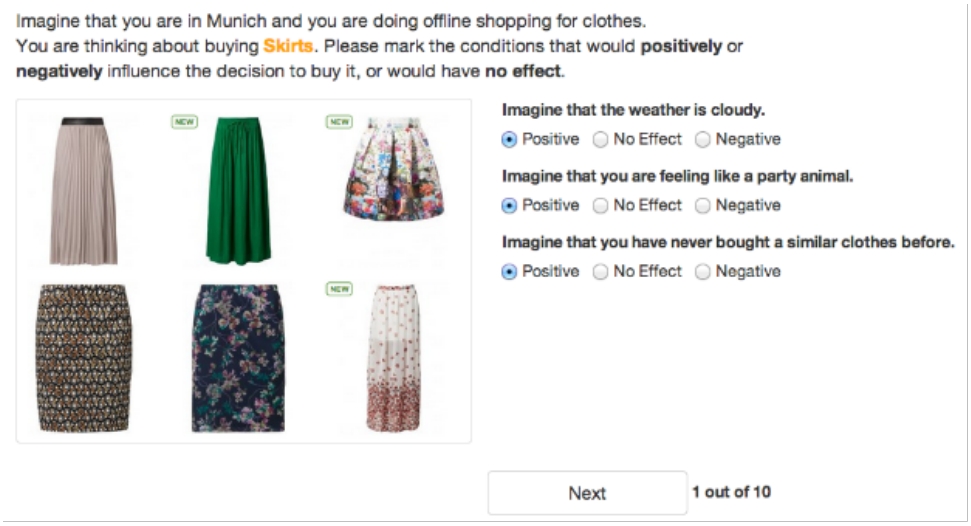


Figure 7.2: Web based survey tool to acquire context relevance

on the intention to buy the particular type of clothing (either 'positive', 'negative' or 'no effect'). An example can be seen in *figure 7.2*. Finally, ten different contextual situations were randomly presented to each user. In total, 38 participants took part in the survey and gave 1190 responses.

Based on the web survey's results we could define samples for the distribution $P(I|C_i, T)$ where I (Influence) is the context's influence variable being assigned to one of the three values: 'Positive', 'negative' or 'no effect', T is the clothing category (e.g., trousers), and $C_1, ...C_N$ are the context factors that may influence the buying decision. This distribution models the influence of the context factors on the user's purchasing decision considering different clothing categories. The spread of a categorical variable $X = x_1, ...x_n$ can be measured by looking at the entropy of the random variable [Baltrunas et al., 2011]. If $P(X = x_i) = \pi_i$, the entropy of X is:

$$H(X) = -\sum_{1 \leq i \leq n} \pi_i \cdot log\pi_i$$

The spread can be used to estimate the association between variable X_1: User's intention to buy a certain item (e.g., positive) and variable X_2: The expected influence of the context factor on the user decision (e.g., the current budget has a high influence on the buying decision of a shirt). For example, if the influence of the context factor is strong, then the spread of variable X_1 will be reduced if the user is aware of her current bank balance, and if the influence is weak, the spread of X_1 remains unchanged even if the bank balance is known. This association can be formally defined as in [Baltrunas et al., 2011]:

$$U = \frac{H(X_1) - H(X_2)}{H(X_1)}$$

where $H(X_1) - H(X_2)$ is the difference between the spread of X_1 and the expected spread of X_2 which measures the influence of a specific context factor to the user's decision.

If U is 1, then the influence is certain for each value of a context factor ($H(X_2) = 0$). On the other hand, U is zero if the context factor does not have any influence on the user decision ($H(X_1) = H(X_2)$). U can be seen as the mutual information of X_1 and X_2 normalized to the interval [0,1] and helps understanding which context factors may decrease the uncertainty about the user decision [Baltrunas et al., 2011]. We compute U for each context factor for all clothing types and use it as a weighting factor for the similarity assessment. The ordered context factors in descending order of U for each clothing category can be seen in *Appendix D.1.*

7.2.2 The Proposed Approach

To integrate the contextual information into the recommender system, context-driven querying and -search was adopted. This approach uses contextual information and/or user's specified interest to query or search a repository of resources and present the most appropriate ones to the user. Corresponding to this approach, we applied a case-based recommendation technique (see *subsection 2.1.4* for further details). Each case in the case base is composed of an item and the contextual situation under which the item is bought. Here a contextual situation is a combination of several context factors and their corresponding values. A user query is composed of a logical query with fixed context constraints and a feature value vector of context factors and their corresponding value that the user wishes to be considered in the recommendations. For example, if a user is a budget buyer and wants to buy sportswear when the temperature is hot in opened stores nearby, the query may be structured as follows:

$$\begin{aligned} query = \{&((distance \leq 2000m) \wedge (timeopen = now + 30min)), \\ &(budget(budget\ buyer), intent(sports), temperature(hot))\} \end{aligned} \tag{7.2}$$

The system then searches the case base and selects the nine cases with the most similar context situation and recommends these items or similar ones to the user. However those items are not only ranked according to the level of similarity to the current context. Previous works have shown that diversity is an important consideration to ensure the coverage of the current scope of candidate items, in particular in exploratory scenarios. Thus, the bounded greedy selection algorithm (presented in [Lamche et al., 2014c]) was extended to select the cases with the most diverse set of items among the retrieved most similar cases. Knowledge-based recommender systems have the disadvantage of static suggestion ability because the knowledge base is usually preset by domain experts and barely changes. We therefore integrated a collaborative filtering approach so that the users can play the expert role and their purchased items together with the contextual information can be added to the case base as a new case for future recommendations [Ricci et al., 2002].

7.2.3 Case Model

The case base consists of two components: The item bought (I) and the context situation (C):

$$CB = I \times C \tag{7.3}$$

Each case $c = (i, e) \in CB$ in the case base is composed of two sub-elements i, e which are instances of the spaces I, C respectively. The cases are not correlated with the user who submits it, thus the cases are not linked to the user model [Ricci et al., 2002]. A case is created when the user purchases the item. C is the data structure that defines the context situation under which the item is bought. It is composed of a feature value vector of context factors and their corresponding values that the user wishes to be considered and a feature value vector of context factors and their corresponding factor importance weights. The factor importance weights reflect the level of influence of the context factors on the recommendations of clothing items. They are determined by the type of clothes and have been calculated in the experiment introduced in *subsection 7.2.1.* For a full list of the factor importance weights for different clothing type see *Appendix D.1.* An example for the feature value vector for a budget buyer who is looking for sports clothes when the temperature is hot could be:

$$\begin{aligned} context_{attributes} = \{&(budget(budgetBuyer), intent(sports), temperature(hot)), \\ &((w_{budget}(0.7), w_{intent}(0.6), w_{temperature}(0.9))\} \end{aligned} \tag{7.4}$$

I is the data structure that describes the clothing item bought by the user. It is represented as a feature value weight vector and was borrowed directly from the baseline system introduced in [Lamche et al., 2014c]. To provide recommendations, cases with context situations similar to the user's current context can be retrieved and the items contained in those cases can be used directly for the recommendations. They can also be used as reference items to find other similar items to recommend.

7.2.4 Similarity Assessment

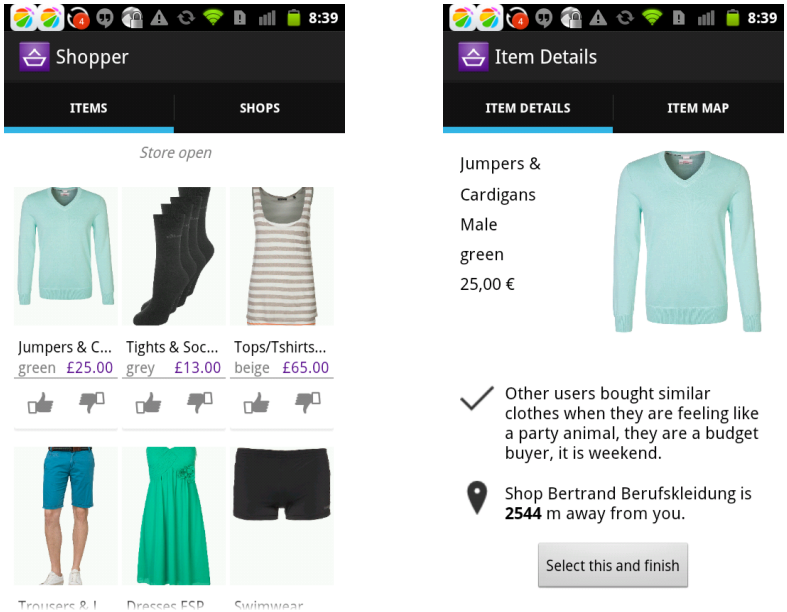
In order to get the similarity between the current context and retrieved cases, the Heterogeneous Euclidean-Overlap Metric (HEOM) was borrowed [Ricci et al., 2002]:

$$heom(x, y) = \frac{\sqrt{\sum_{i=1}^{n} w_i d_i(x_i, y_i)^2}}{\sqrt{\sum_{i=1}^{n} w_i}}$$

$$where : d_i(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \text{ or } y_i \text{ are unknown} \\ overlap(x_i, y_i) & \text{if the i-th feature is symbolic} \\ \frac{|x_i - y_i|}{range_i} & \text{if the i-th feature is finite integer or real} \end{cases} \tag{7.5}$$

Here $range_i$ is the difference between the maximum and minimum value of a numeric feature, and $overlap(x_i, y_i) = 1$ if $x_i \neq y_i$ and 0 otherwise. The weights $0 \leq w_i \leq 1$ correspond to the weighting factors, which have been calculated in the experiment introduced

before. This metric measures the distance between two vectors. The further away two vectors are, the higher the similarity. By using the previously discussed case model and query structure, the feature value vectors of context factors describing the context situation in both structures can be fed into the similarity metric. After the similarities between the submitted query and the retrieved cases are calculated, the cases will be ranked according to the calculated similarity. Then the bounded greedy selection algorithm presented in [Lamche et al., 2014c] is used to select and rank the cases based on the diversity of items contained in those cases, which are then presented to the user (see *figure 7.3* for the corresponding user interfaces).



(a) Set of recommendations (b) Detail view of an item

Figure 7.3: Final design of the recommendation interface

7.3 User Study

The primary goal of the evaluation is to find out whether the users perceive a difference in the accuracy of recommendations when comparing a context-aware recommender system with a system not considering context. We therefore measure the perceived accuracy. A second goal of the study is to find out whether users are more satisfied with a recommender system that takes the mobile context into account. We therefore ask the users to overall rate both systems. Since the main goal of our developed conceptual framework is an improved user experience of mobile recommender systems, we also evaluate the user's perceived effort and measure the consumed time and number of critiquing cycles needed to finish a task.

	CARS		Baseline		p value
	mean	stdev	mean	stdev	
Perceived Effort	3.74	0.92	3.61	1.16	.3
Perceived Accuracy	4.09	0.79	3.74	0.92	**.029**
System Preference	4.04	0.77	3.39	0.94	**.004**
Time Consumption	122.91 s	77.67	117.52 s	73	.405
Critiquing Cycle	2.83	2.46	3.43	2.35	.171

Table 7.1: Overview of the evaluation result

7.3.1 Setup

Overall a number of 23 people participated in the user study. The study was designed as within-subjects, one group of people tested both variants: The system introduced in the previous sections (CARS) and a baseline, which basically recommends items without eliciting initial preferences from the user and also uses a diversity-based approach to ensure the coverage of the recommended items. The baseline was presented in [Lamche et al., 2014c]. The order of which system was tested first was alternated between subjects. The user was asked to select the most appealing item while imagining herself being in a context scenario that was randomly selected by the system from a set of five pre-created context scenarios. A typical context description was: *"Imagine that you want to buy clothes for daily wear, you are a budget buyer and the temperature is cold. You don't care about the distance to the shops."* To see the full list of scenarios, refer *Appendix D.2*. For each context scenario, about four different context factors were included. After the users tested both variants, they were asked to fill out a survey (see *Appendix D.3*). An overview of the evaluation results can be seen in *table 7.1*. Next to the means of the two systems, the standard deviations are shown and the last column denotes the p-value of a one-tail paired t-test with 22 degrees of freedom (23 participants - 1) at a 0.05 level of significance. Results that are statistically significant are printed bold. The testing framework used for the evaluation follows the one presented in [Chen and Pu, 2009]. In particular we measured: *Perceived effort, perceived accuracy, system preference, time consumption* and *number of critiquing cycles*. More details about the evaluation technique can be found in *section 2.3*. All perceptive measures were gathered through a questionnaire listing statements that had to be rated on a 5-point Likert scale (from 1, strongly disagree to 5, strongly agree with 3 being neutral). The time consumption was stopped in seconds and the number of critiquing cycles was automatically counted. The final results will now be discussed in more detail.

7.3.2 Results

To measure the perceived effort of the system, the user was asked if it was easy to find the required information. When looking at the mean of the rate of this question in *table 7.1*, the context-aware system slightly beats the baseline (3.74 vs. 3.61). However the difference is not significant ($p = 0.3$). See *figure 7.4* for an illustration.

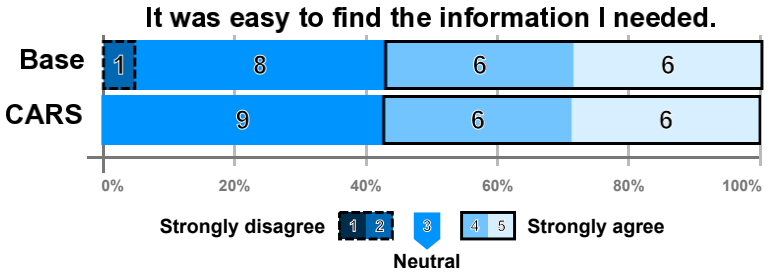


Figure 7.4: Distribution of ratings for perceived effort on a 5-point Likert scale

To measure the accuracy of the system, the user was asked if the system provided accurate recommendation in order to help completing the scenario. The accuracy of the context-aware system was rated significantly better than the baseline ($p = 0.029$). Some users mentioned that they found the context settings quite useful during the test. The average rate of the context-aware system is also higher (4.09) than the baseline (3.74), such as illustrated in *figure 7.5*.

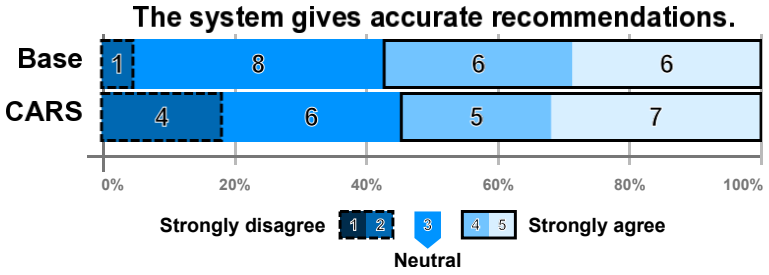


Figure 7.5: Distribution of ratings for perceived accuracy on a 5-point Likert scale

Users were also asked to rate how much they liked using these two systems. In *table 7.1*, it can be seen that the mean rate of the context-aware system is higher than the one of the baseline by about 0.65 points and the difference is significant (with $p = 0.004$). See *figure 7.6* for an illustration.

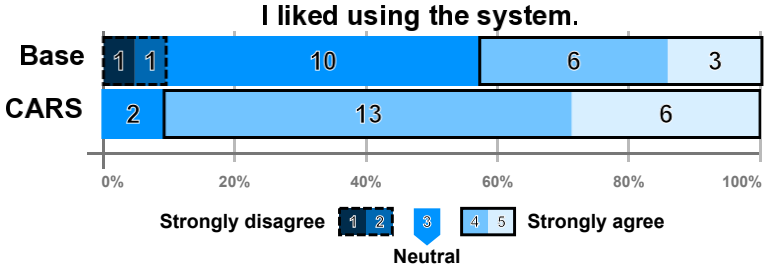
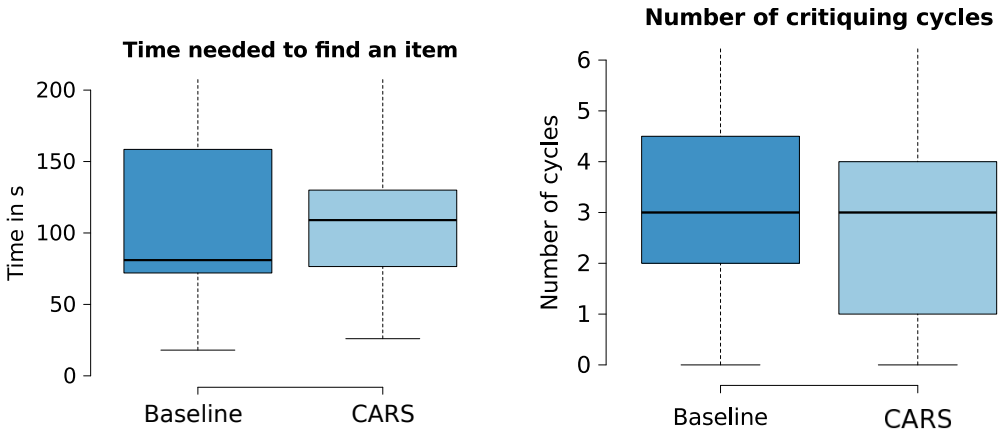


Figure 7.6: Distribution of ratings for the user's overall rating of the two systems on a 5-point Likert scale

The baseline slightly beats the context-aware system in terms of time consumption when looking at the average time in seconds (117.52 vs. 122.91), but not significantly (see *figure 7.7a*). However, the majority of time consumption of the context-aware variant is more stable and is neither too long nor too short.

The number of critiquing cycles of the context-aware recommender system is on average smaller compared to the baseline. The mean of critiquing cycles of the context-aware system (2.83 cycles) is also smaller than the one of the baseline (3.43 cycles), but the difference is not significant ($p = 0.171$). See *figure 7.7b* for an illustration.



(a) Required time to find an item

(b) Number of critiquing cycles to find an item

Figure 7.7: Box plots of the time in seconds (left) required to complete a session and number of critiquing cycles (the maximum has been omitted for space reasons)

To summarize the results of this user study, we were able to show that our developed context-aware system has a better performance regarding prediction accuracy and decision effort. The users considered the clothing items provided by the context-aware system as more appropriate compared to the baseline. The context-aware system was also perceived as being significantly more accurate in delivering personalized recommendations to the users' different context scenarios. Furthermore, users showed a clear preference of the context-aware variant and were able to understand the benefits of taking contextual information into account very well. We can conclude that the proposed context-aware recommendation approach is able to deliver more accurate recommendations compared to a recommender system that does not consider context. Hence, the practical advantage of the proposed recommendation approach can be indicated.

7.4 Conclusion and Next Steps

In this chapter, a context-aware recommender system was developed and evaluated in a mobile shopping scenario. The system is based on an Active Learning approach and uses a nearest neighbor algorithm. We first acquired the relevance of contextual factors by asking users to rate the influence of different context factors on their purchasing decisions. For this purpose, we developed a web tool that presents the context scenarios to the users and allows them to rate a specific context factor by imagining themselves being in the selected situation. After the mobile context relevance was obtained, a case-based recommendation approach was proposed to integrate contextual information into the recommender system by recommending clothing items bought by other users being in a similar contextual condition. We integrate a collaborative filtering approach and add the user's selected items together with the contextual information to the case base as a new case for future recommendations. To compute the similarity between a retrieved case and a user's submitted query, the Euclidean Overlap Metric (HEOM) was applied. We then evaluated an Android mobile application using the developed concept within a user study. Results show that our presented CARS performs better regarding prediction accuracy and decision effort. The users consider the clothes items provided by the CARS as being more appropriate compared to a baseline system that does not consider context. The CARS is also perceived to be significantly more accurate in providing personalized recommendations for the user's different context scenarios. Furthermore, users show a clear preference of the CARS compared to the baseline and understand the benefits of a system taking contextual information into account very well. The results of this chapter transform our developed mobile recommender system into a context-aware one. It consists of a critique-based recommendation algorithm, a stereotype user model, evaluated interaction methods and now also takes mobile context into account. However, mobile explanations of the recommended items remain still an open research topic. Explanations are nowadays considered as important for recommender systems because they increase, among other things, scrutability and trust in the system. Results of the online survey conducted in *chapter 6* show that users appreciate explanations of mobile recommendations (see also *Appendix C.1* for an overview of the user study results). The following chapter will therefore focus on explanations of mobile recommender systems and investigate how they can be automatically generated.

8

Explanations

This chapter focuses on explanations of mobile recommendations. Explanations of recommendations help users to make better decisions in contrast to recommendations without explanations, e.g., by increasing the transparency between the system and the user. There are two main goals of this chapter. One is to study whether a mobile recommender model with interactive explanations leads to more user control and transparency in critique-based mobile recommender systems. Second is to develop a strategy to generate interactive explanations in a content-based recommender system. A mobile application is developed and evaluated by following the proposed concept. We first start off with some definitions relevant for explanations in recommender systems and summarize related work (1). The next section (2) explains the reasoning behind and the path towards integrating interactive explanations into a mobile recommender system. The user study evaluating the developed system is discussed in the following section (3). We close by suggesting opportunities for future research (4).

8.1 Motivation

The feedback of the user study of *chapter 4* and of the online survey of *chapter 6* showed that mobile users appreciate explanations so that the logic of the recommender system can be reproduced. Even very popular recommender systems, such as the e-commerce company *Amazon* might lead to sometimes obscure results. For example, if a user recently bought a refrigerator, *Amazon.com* keeps suggesting other ones, but almost no one needs a second refrigerator within a short time. Interactive explanations might solve this problem. The system justifies its decision and allows the user to add missing information, or in case of wrong assumptions, correct those. Moreover, explanations of recommendations help users to make better decisions in contrast to recommendations without explanations while also exposing the reasoning behind a recommendation [Tintarev and Masthoff, 2012]. Recommender systems employing explanations so far did not leverage their interactivity aspect.

Touch based interfaces in smartphones reduce user effort while giving input. This can empower the interactivity for explanations and at the same time increase the user control and transparency. We therefore develop a new way of explaining recommendations, customized to a mobile device where display size is limited. A mobile shopping recommender system applying those strategies will be implemented and tested for its advantages and drawbacks. Our main goal is to investigate whether our approach to automatically generate interactive explanations has a positive effect on the system's transparency and user control.

8.1.1 Explanations in Recommender Systems

Recommender systems have become a popular and powerful tool to help users to find suitable items fast and with less effort. While the general idea of explaining recommendations already exists for a long time now and early studies have shown their benefits (e.g., [Herlocker, 1999]), most of the recommender systems are still black boxes which neither expose how recommendations were created nor do they contain methods to guide the exploring process. However, the algorithms are getting more and more complex, often combining different strategies (see *hybrid recommender systems* in section 2.1.4). Also the integration of information from social networks as well as mobile context into the recommendation process makes it for the users even harder to understand the reasoning behind. Generating interactive explanations for mobile recommender systems therefore seems to be important in order to allow a positive user experience. Tintarev et al. define the following seven goals for explanations in recommender systems [Tintarev and Masthoff, 2012]:

Transparency to help user's understand how the recommendations are generated and how the system works. It allows users to check the quality of the system and in case of anomalies it lets users understand why the system has reached a surprising result.

Scrutability to help users correct wrong assumptions made by the system. As recommender systems collect information in the background and change their internal state accordingly, recommendations might not always match the user's current preferences. Therefore, it is important that explanations enable the users to understand how the system works and let them exert control over the type of recommendations [Sø rmo et al., 2005].

Trust to increase users' confidence in the system. Chen and Pu consider the perceived competence of a system as the main positive influence of building trust [Chen and Pu, 2005]. Grabner-Kräuter and Kaluscha see trust as being able to reduce the complexity of human decision making when they have to deal with uncertainty [Grabner-Kräuter and Kaluscha, 2003]. The first part is often linked with transparency, as understanding a system leads to higher error tolerance. The second definition focuses on helping a user to accept a recommendation. Increasing trust is also important when for instance private data is used.

Persuasiveness to convince users to try or buy items and enhance user acceptance of the system. This means that the explanations have to be designed in a way so that the

recommendation gets accepted. This can be done by adding specific arguments that might convince the user to buy the product [Jannach et al., 2010].

Effectiveness to help users make better decisions. A system with effective explanations would let users discover their preferences and support them to make decisions they actually end up liking [Jannach et al., 2010]. Tintarev and Masthoff highlighted the importance of personalization to the individual user, as well as other factors such as the source of recommendations, user mood, the effect of group viewing and the effect of explanations on user expectations [Tintarev and Masthoff, 2007b].

Efficiency to help users decide faster, which recommended item is the best for them. The goal of efficiency is to reduce effort, which can be split into three different aspects. The first one is the time needed to make a decision. The second one is the number of cycles or items recommended and the last aspect is the perceived cognitive effort of the user. A *conversational recommender system*, like the one implemented in this work, can be considered to already implicitly contain explanations targeting efficiency in form of critiquing dialogs [Tintarev and Masthoff, 2011], [Chen and Pu, 2009].

Satisfaction to increase the user's satisfaction with the system. Presence of longer descriptions of individual items has been found to be positively correlated with both the perceived usefulness and ease of use of the recommender system [Sinha and Swearingen, 2002]. Tintarev and Masthoff argue that this can be seen as increased overall satisfaction [Tintarev and Masthoff, 2007a].

However, meeting all these criteria is unlikely, some of these aims are even contradicting such as persuasiveness and effectiveness. Thus, choosing which criteria to improve is a trade-off.

Explanations might also differ by the degree of personalization. While non-personalized explanations use general information to indicate the relevance of a recommendation, personalized explanations clarify how a user might relate to a recommended item [Tintarev and Masthoff, 2012]. To achieve the goals mentioned before, explanations have to be planned and designed thoroughly. There is a strong connection between explanations and the underlying algorithm of a recommender system. Many key goals of the explanation require the users understanding of the basic concepts used in the system. Therefore it is not surprising that explanations often use the algorithms input values in order to generate the explanations (e.g., a content-based recommender system could explain a recommendation with "*this item is recommended because you like black dresses*") [Tintarev and Masthoff, 2011].

8.1.2 Existing Approaches

Due to the benefits of explanations in mobile recommender systems, a lot of research has been conducted in this context. Since our work focuses on explanations that aim

at improving transparency and scrutability in a recommender system, we investigated previous research in these two areas.

The work of Vig et al. separates justification from transparency. While transparency should give an honest statement of how the recommendation set is generated and how the system works in general, justification can be derived from the recommendation algorithm and explain why a recommendation was selected. Vig et al. developed a web-based *Tagsplanations* system where the recommendation is justified using relevance of tags [Vig et al., 2009]. An example for such an explanation is *"we recommend the movie Fargo because it is tagged with quirky and you have enjoyed other movies tagged with quirky"* [Vig et al., 2009, p. 2]. Their approach, as the authors noted, lacked the ability to let users override their inferred tag preferences.

Cramer et al. applied transparent explanations in the web-based *CHIP* (Cultural Heritage Information Personalization) system that recommends artworks based on the user's ratings of artworks. The main goal of the work was to make the criteria more transparent, the system uses to recommend artworks. It did so by showing the users the criteria on which the system based its recommendation. The authors argue that transparency increased the acceptance of the system [Cramer et al., 2008].

An interesting approach to increase scrutability has been taken by [Czarkowski, 2006]. The author developed *SASY*, a web-based holiday recommender system which has scrutinization tools that aim not only to enable users to understand how the system works, but also to let them take control over recommendations by enabling them to modify data that is stored about them.

TasteWeights is a web-based social recommender system developed by Knijnenburg et al. aiming at increasing inspectability and control. The system provides inspectability by displaying a graph of the user's items, friends and recommendations. The system enables control over recommendations by allowing users to adjust the weights of the items and friends they have. The authors evaluated the system with 267 participants. Their results showed that users appreciated the inspectability and control over recommendations. The control given via weighting of items and friends made the system more understandable. Finally, the authors concluded that such interactive control results in scrutability [Knijnenburg et al., 2012].

Wasinger et al. apply scrutinization in a mobile restaurant recommender system named *Menu Mentor*. In this system, users can see the personalized score of a recommended restaurant and the details of how the system computed that score. However, users can change the recommendation behavior only by critiquing presented items via meal star ratings and no granular control over the meal content is provided. A conducted user study showed that participants perceived enhanced personal control over given recommendations [Wasinger et al., 2013].

Netflix is one of the biggest online movie streaming portals and has recently also entered the german market. On *Netflix*, it is not just possible to stream different movies or series, but users can also set up a user profile. When using *Netflix* for the first time, the customer

is required to select five movies she likes. At a later stage, movies can be rated on a scale from 1 to 5. Furthermore, users can add information about how often they watch different genres and how much they like them. It is also possible to connect with *Facebook*, so preferences of friends can be considered as well. Another important element in *Netflix'* personalization is awareness of the users to gain trust and encourage them to give more feedback. This shall also be supported by providing explanations which declare why the system decided to recommend a certain movie or show [Amatriain, 2013].

In summary, although previous research focused on increasing either scrutability or transparency in recommender systems, no research was conducted on how interactive explanations can increase the user experience in mobile recommender systems. The difference when developing interactive explanations for a web-based system compared to a mobile one is not only the display size but also different interaction methods that need to be applied. Moreover, mobile recommender systems should take specific information such as the current location into account in order to create added value for the user. In particular we think that interactive explanations have the potential to increase scrutability, ability to trust, transparency as well as persuasiveness in a mobile recommender system.

8.1.3 Goals

Our system aims at offering shoppers a way to find nearby shopping locations with interesting clothing items while also supporting them in decision making by providing interactive explanations. Mobile recommender systems use a lot of situational information to generate recommendations, so it might not always be clear to the user how the recommendations are generated. Introducing transparency can help solving this problem. However, mobile devices require even more considerations in the design and development, e.g., due to the small display size. Thus, different interaction methods should also be taken into account when generating transparent explanations. Moreover, the explanation framework should generate textual explanations that make it clear to the user how her preferences are modeled. In order to not bore the user, explanations should be concise and include variations in wording. Furthermore, introducing transparency alone might not be enough because users often want to feel in control of the recommendation process. The explanation goal scrutability addresses this issue by letting users correct system mistakes. There have been several approaches to incorporate scrutable explanations to traditional web-based recommender systems. However, more investigation is required in the area of mobile recommender systems. First of all, the system should highlight the areas of textual explanations that can be interacted with. Second, the system should allow the user to easily make changes and get new recommendations. While transparent and scrutable explanations are the main focus of this work, there are also some side goals, such as satisfaction and efficiency. We design a test application implementing such an approach to investigate the following questions:

Research Questions: What type of explanations methods should be applied? Which explanation styles are more effective in terms of helping users in decision making and which are better in terms of transparency? How does a solution appropriate for a device with limited screen size look like? Will the developed system increase the overall user experience?

8.2 Designing the Test Application

The aim of the prototype is to study whether a mobile recommender model with interactive explanations leads to more user control and transparency in critique-based mobile recommender systems. For this purpose we develop a prototype that automatically generates interactive explanations for recommendations in a mobile shopping scenario and compare it to a baseline without interactive explanations. The concept, the implementation and the evaluation of our approach to generate interactive explanations have been published in [Lamche et al., 2014a]

8.2.1 How Explicit Feedback Affects Weights

Our system uses two types of user feedback. One of them enables critiquing the recommended items on their features (which was already provided in our baseline system, described in *chapter 4*). The other feedback strategy allows correcting mistakes regarding the user's preferences via explicit preference statements. Explanations are designed to be interactive, so that the user can state her actual preference over feature values after tapping on the explanation. If the user states interests on some feature values, a new value vector will be initialized for the query with all interested values being assigned equal weight summing to 1.0 and the rest having 0.0 weight. Thus the system focuses on the stated feature values, whereas the other values will be avoided. For example if a user interacts with the explanation associated with this query:

$$color_{red,blue,green,white,black}(0.2, 0.2, 0.2, 0.2, 0.2)$$

and states that she is actually only interested in blue and green, then the resulting new weight vector would look like the following (which will influence the search query and thus the new recommendations):

$$feedback_{positive}(blue, green) : color_{red,blue,green,white,black}(0, 0.5, 0.5, 0, 0)$$

8.2.2 Generating Interactive Explanations

The main vision behind interactive explanations is to use them not only as a booster for transparency and understandability of the recommendation process but also as an enabler for user control. In order to explain the current state of the user model (which stores the user's preferences) and the reasoning behind recommendations, two types of explanations are defined: Interactive *recommendation-* and *preference* explanations.

Interactive Recommendation Explanations

Interactive *recommendation explanations* are textual explanations with two aims: First they justify why an item in the recommendation set is relevant for the user. Second they let the user make direct changes to her inferred preferences. The generation is based on the set of recommended items, the user model and the location.

Argument Assessment: The argument assessment method is used to determine the quality of every possible argument about an item. It is based on the method described by Bader et al. It uses Multi-Criteria Decision Making Methods (MCDM) to assess items I on multiple decision dimensions D (e.g., features that an item can have) by means of utility functions. Dimensions in the context of this recommender system are features and the location of the user. The method described in [Bader et al., 2011] uses four scores, which lay a good foundation for the method in this work. However, their calculations have to be adapted to the underlying recommendation infrastructure to produce meaningful explanations.

Local score: The local score $LS_{I,D}$ measures the performance of a dimension without taking into account how much the user values that dimension. Our system uses feature value weight vectors to consider both item features and features in a query, which represents the current preferences of the user. A feature's local score is the scalar product of the weight vector (for that feature) in the query with respective weight vector in the item's representation. It is formalized as below, where $w_{I,D}$ represents the feature value weight vector for item dimension D and $w_{Q,D}$ represents the feature value weight vector for query dimension D and n stands for the number of feature values for that dimension:

$$LS_{I,D} = \sum_{i=0}^{n-1} w_{I,D}(i).w_{Q,D}(i) \tag{8.1}$$

Explanation score: The explanation score $ES_{I,D}$ describes the explaining performance of a dimension. The weight for each dimension is calculated dynamically by using a function that decreases the effects of the number of feature values in each dimension. It is formalized as follows, where $length_{w_D}$ denotes the number of feature values in a specific dimension D and $length_{total\_attribute\_values}$ the total number of feature values for all dimensions. Using the square root produced good results since it limits the effect of a high amount of feature

values (e.g., if the dataset consists of hundreds of different brands) on the calculation of weights.

$$w_D = \sqrt{\frac{length_{w_D}}{length_{total\_attribute\_values}}} \tag{8.2}$$

With the following dynamically calculated weight for a dimension, the explanation score of the dimension can be calculated by multiplying it with the local score of that dimension:

$$ES_{I,D} = LS_{I,D}.w_D \tag{8.3}$$

Information score: The information score IS_D measures the amount of information provided by a dimension. The calculation of the information score suggested in [Bader et al., 2011] is preserved as it already lays a good foundation to reason whether explaining an item from a given dimension provides a good value. So, it can be defined as follows where R denotes the range of explanation scores for that dimension for all recommended items and I denotes the information that dimension provides for an item:

$$IS_D = \frac{R + I}{2} \tag{8.4}$$

Range R is calculated as the difference between the maximum and minimum explanation score for the given dimension for all recommended items, namely $R = max(ES_{I,D}) - min(ES_{I,D})$. Information I, however, is calculated quite differently from the strategy proposed by [Bader et al., 2011]. In their system, a dimension provides less and less information as the number of items to be explained from the same dimension increases. This does not apply to the context of the clothing recommender system developed for this work. An item could still provide good information if there are not so many items that can be explained from the same feature value. For instance, it is still informative to explain an item from the color blue; although another item is also explained by the same dimension (color) but from a different value, let's say green. Therefore, I is calculated as a function of the size of recommendation set (n) and number of items in the set that has the same value for a dimension (h): $I = \frac{n-h}{n-1}$.

Global score: The global score GS_I measures the overall quality of an item in all dimensions. It is the mean of explanation scores of all of its dimensions. The following formula demonstrates how it is formalized, where n denotes the total number of all dimensions and ES_{I,D_i} the explanation score of an item on i_{th} dimension.

$$GS_I = \frac{\sum_{i=0}^{n-1} ES_{I,D_i}}{n} \tag{8.5}$$

The above-defined methods for calculating explanation and information scores are only valid for item features. Explanations should also take the user's current location into account. The explanation score of the location is calculated using domain knowledge. More precisely, the explanation score is inversely proportional to the distance between the current location of the user and the shop where the explained item is sold. The explanation score gets higher as the distance gets lower. The information score is calculated with the same formula defined earlier for features $IS_D = \frac{R+I}{2}$, but Information I slightly changes. As proposed earlier, it is calculated using the formula $I = \frac{n-h}{n-1}$, but in this case h stands for the number of items with similar explanation score.

Argument Types: In order to generate explanations with convincing arguments, different argument aspects are defined by following the guidelines for evaluative arguments described in [Carenini and Moore, 2006]. Moreover, the types of arguments described in [Bader et al., 2011] are taken as a basis. First of all, arguments can be either *positive* or *negative*. While positive arguments are used to convince the user to the relevance of recommendations, negative arguments are computed so that the system can give an honest statement about the quality of the recommended item. The second aspect of arguments is the type of dimension they explain, *feature* or *location*. Lastly, they can be *primary* or *supporting* arguments. Primary arguments alone are used to generate concise explanations. Combination of primary and supporting arguments are used to generate detailed explanations. We distinguish between five argument types: *Strong primary feature arguments*, *Weak primary feature arguments*, *Supporting feature arguments*, *Context arguments* and *Negative arguments* (which indicate that the user is actually not interested in that value).

Explanation Process: The explanation process is based on the approach described in [Bader et al., 2011] but it is adapted to use the previously defined argument types. Different from the system of Bader et al., explanations are designed to contain multiple positive arguments on features. Negative arguments are generated but only displayed when necessary by using a ramping strategy. *Figure 8.1* shows the process to select arguments. It follows the framework for explanation generation described in [Carenini and Moore, 2006] as the process is divided into the selection and organization of explanation content and the transformation in a human readable form.

Content Selection: The argumentation strategy selects arguments for every item I separately. One or more primary arguments are selected first to help the user to instantly recognize why the item is relevant. There are four alternative ways to select the primary arguments (alternatives 1 to 4 in *figure 8.1*). The first alternative is that the item is in the recommendation set because it was the last critique and it was carried (1). Another reason might be that the system has enough *strong arguments* to explain an item (2). If there are not any strong arguments, the strategy checks if there are any *weak arguments* (3). In case there are one or more weak arguments, the system also adds supporting arguments to make the explanation more convincing. Finally, if there are no weak arguments too, then the item is checked if it is a good average by comparing its global score GS_I to threshold β (4). If so, similar to alternative (3), supporting arguments are also added to increase the

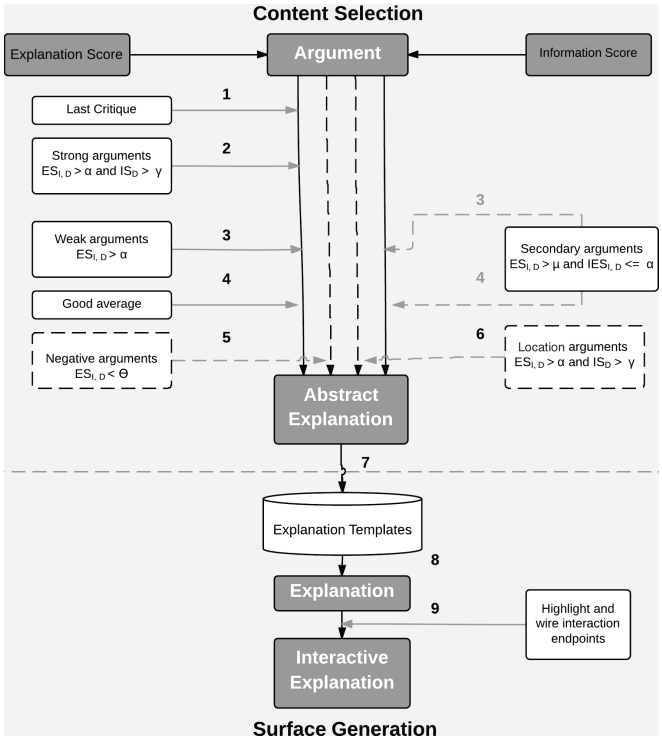


Figure 8.1: Generation of explanations

competence of the explanation. Otherwise the strategy supposes that the recommended item is serendipitous and added to the set to explore the user's preferences. With one or more primary arguments, the system checks if there are any negative arguments and context arguments to add (5 and 6).

Surface Generation: The result of the content selection is an *abstract explanation*, which needs to be transformed into something the user understands. This is done in the surface generation phase. Various explanation sentence templates are decorated with either feature values or context values (7 and 8). Explanation templates are sentences with placeholders for feature and location values stored in XML format. The previously determined primary argument type is used to determine which type of explanation template to use. Feature values in the generated textual output are then highlighted and their interaction endpoints are defined (9). The resulting output is a textual explanation, highlighted in the parts where feature values are mentioned. They are interactive such that, after the user taps on the highlighted areas, she can specify what she exactly is looking for.

Interactive Preference Explanations

Interactive preference explanations have two main goals. First, they aim at letting the user inspect the current state of the system's understanding of the user's preferences. Second, they intend to let the user make direct changes to the preference. Two main types of

preferences explanations are defined, *interactive textual explanations* and *interactive visual explanations*.

Generating Textual Preference Explanations: The only input to textual preference explanation generation algorithm is the user model. For each dimension D the algorithm can generate interactive explanations. Dimensions are features that an item can have. The algorithm distinguishes between four feature value weight vectors, indicating different user preferences: First, the user is indifferent to any feature value. Second, the user is only interested in a set of feature values. Third, the user is avoiding a set of feature values. And fourth, the user prefers a set of feature values over others.

Generating Visual Preference Explanations: Visual preference explanations are also generated by using the user model, more specifically by making use of the array of feature value weight vectors, which represents the user's current preferences. For each feature, there is already a feature value weight vector, which indicates the priorities of the user among feature values. All those weights are between 0.0 and 1.0 summing up to 1.0. They could be scaled to a percentage to generate pie charts illustrating numerical proportion. *Figure 8.5* illustrates this chart representation.

Using Text Templates Supporting Variation

XML templates are used to generate explanation sentences for the different user preference types. Those templates contain placeholders for feature and context values which are replaced during the explanation generation process. For *recommendation explanations*, there are a few sentence variations for almost every type of arguments. These templates can be used in combination with each other. For example, *supporting arguments* can support a weak argument. In such cases, argument sentences are connected using conjunctions. See *table 8.1* for examples of the different text templates for recommendation explanations:

Text template	**Example phrase**
Strong argument	*"Mainly because you currently like X."*
Weak argument	*"Partially as you are currently interested in X."*
Supporting argument	*"Also, slightly because of your current interest in X."*
Location	*"And it is just Y meters away from you."*
Average item	*"An average item, but might be interesting for you."*
Last critique	*"Kept so that you can keep track of your critiques."*
Serendipity	*"This might help us discovering your preferences."* or *"A serendipitous item that you perhaps like."*
Negative argument	*"However, it has the following feature(s) you don't like: X, Y [...]."*

Table 8.1: Text templates for recommendation explanations

A similar mechanism is also used for *preference explanations*. However, to keep it simple, we do not provide variation as the number of features to explain is already limited. See *table 8.2* for selected examples of text templates for preference explanations:

Text template	**Example phrase**
Only some values	*"You are currently interested only in X, Y [...]."* The word "only" in the text is emphasized in bold.
Avoiding some values	*"You are currently avoiding X, Y [...]."* The word "avoiding" is emphasized in bold.
Preferably some values	*"It seems, you currently prefer X, Y [...]."*
Indifferent to feature	*"You are currently indifferent to X feature".*

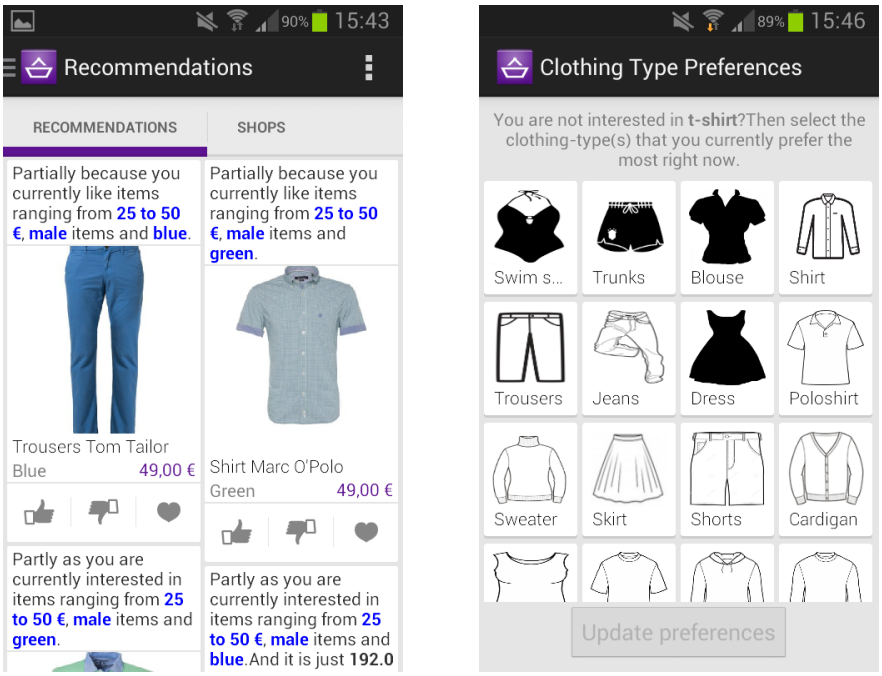
Table 8.2: Text templates for preference explanations

8.2.3 Interaction and Interface Design

The first issue was to clarify how to integrate the interaction process into textual explanations. It was envisioned to give the user the opportunity to tap on the highlighted areas of the explanation text to state her actual preferences on a feature. This leads to a two-step process. First, the user sees an item with an explanation including highlighted words (highlighted words are always associated with a feature, see *figure 8.2a*) and taps on one of them (e.g., in *figure 8.2b*, "t-shirt" was tapped). Then the system directs the user to the screen where the user can make changes. In this second step, the user specifies which feature values she is currently interested in. Based on the stated preferences, the system updates the list of recommendations which completes a recommendation cycle. Note that the critiquing process and associated screens from the baseline (see *chapter 4*) are also implemented in the new approach. Eventually, the interaction strategy consists of critiquing and explicitly stating current preferences. On top of each explicit feedback screen, a text description of what the system expects from the user is given.

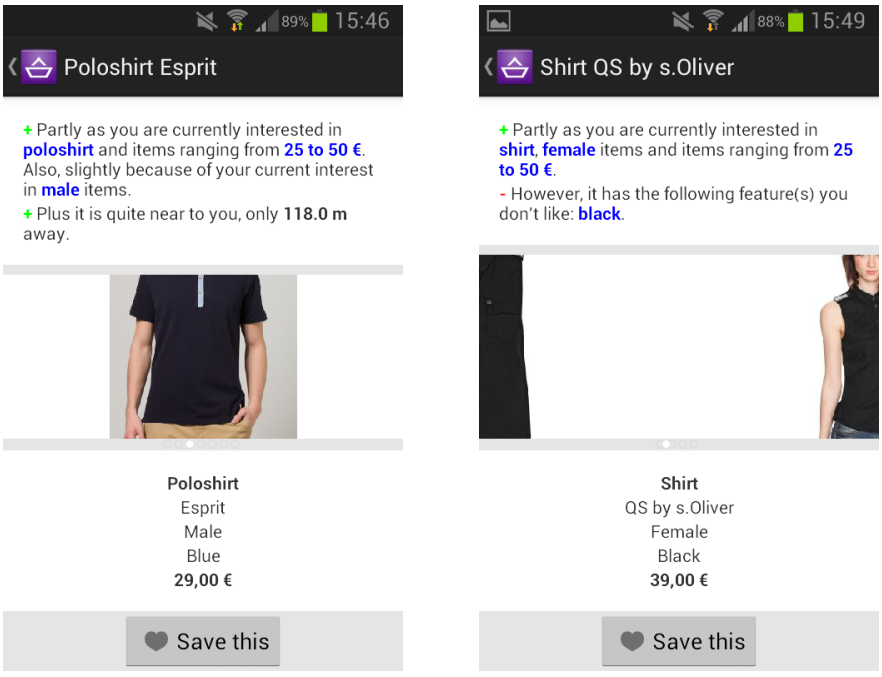
Due to the applied ramping strategy mentioned in *section 8.2.2*, all additional explanations that are less important are not shown as explanations in the list of recommendations but in the screen where more detailed information about the item is presented. Tapping on an item picture accesses that screen. Here, the user can also browse through several pictures of an item by swiping the current picture from right to left (see *figure 8.3b*). In order to make it obvious for the user, the sentences with positive arguments always start with a green "+" sign. Negative arguments, on the other hand, always start with a red "-" sign (see *figure 8.3*).

The next issue was to implement preference explanations, what we call *Mindmap features*. A mindmap feature is the way the system explains its mental map about the preferences of the user. The overview screen for mindmap was designed to quickly show the system's assumptions about the user's current preferences. To keep it simple but yet usable, only textual explanations are used for each feature (see *figure 8.4b*). In order to



(a) Set of recommendations (b) Clothing type adjustment view

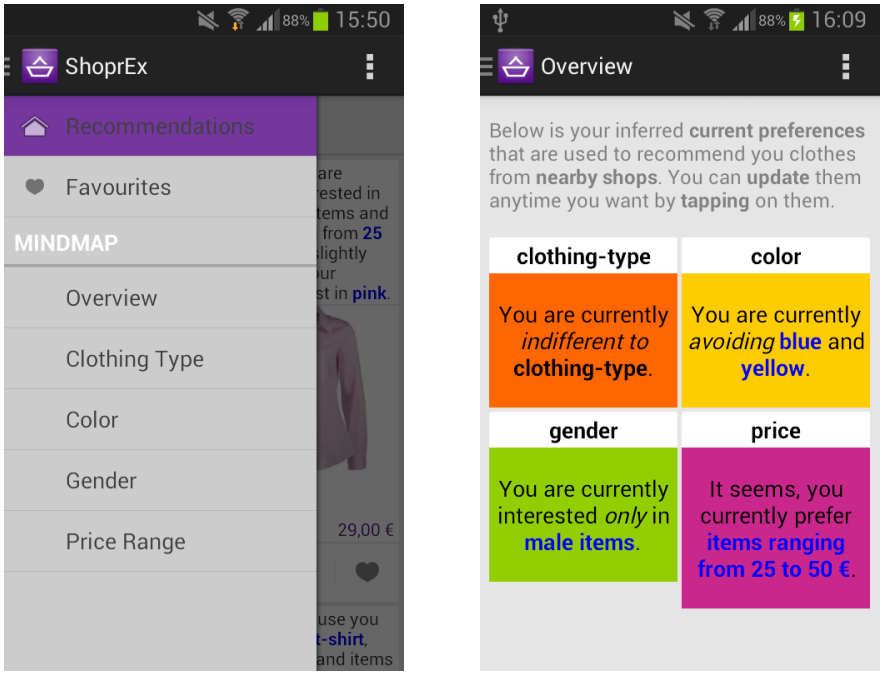
Figure 8.2: Interactive recommendation explanations



(a) Detailed information view (b) Swiping of pictures illustration

Figure 8.3: Detailed information of items

make it easy for the user to reproduce the system's assumptions, the feature values used in the explanation text are highlighted. Moreover, every element representing a feature is made interactive. This lets the user access the explicit feedback screen to state her actual preferences.



(a) Navigation drawer (b) Preference overview

Figure 8.4: Interactive preference explanations

The user should also be able to get a quick overview of the features considered by the system. In order to achieve that, a different "drill down" screen for all screens was developed as part of the mindmap feature. *Figure 8.5* shows the mindmap detail screen for the clothing color feature. The user's preferences on feature values are represented as a chart. Every feature value is displayed as a different color in the charts. One of the most important features is that the highlighted parts of the explanation texts and the charts are interactive as well which lets the user again access the explicit feedback screen.

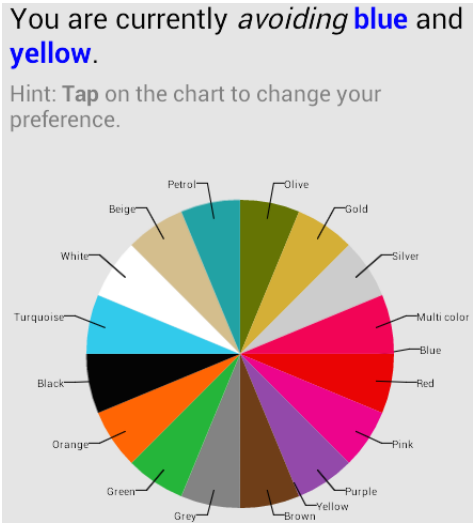


Figure 8.5: Color detail screen

8.3 User Study

Within the evaluation of this prototype we want to find out whether transparency and user control in a mobile recommender system can be improved by feature-based personalized explanations and scrutable interfaces. We also want to investigate whether our approach has a positive effect on the user's satisfaction and the system's efficiency is at the same time not damaged.

8.3.1 Setup

The *test hardware* is a 4.3 inch 480 x 800 resolution Android smartphone (Samsung Galaxy S2) running the *Jelly Bean* version of the Android operating system (4.1.2). Two variants of the system are compared to each other. In order to refrain from the effects of different recommender algorithms, both variants use the same recommendation algorithm which uses diversity-based Active Learning [Lamche et al., 2014c]. Moreover, the critiquing process and the user interface showing more detailed information about an item are exactly the same. The difference of the two tested systems lies in the explanations: The *EXP* variant refers to the proposed system, described in the previous section. In order to test the value of the developed explanations and scrutinization tools, a baseline is needed. As baseline serves the mobile recommender system presented in *chapter 4*. The explanation strategy used in this system is very simple and non-interactive. An explanation text is put on top of all items, which tries to convey the current profile of the user's preferences. It allows the user to observe the effect of her critiques and to compare the current profile against the actually displayed items. An example for such an explanation text is *"avoid grey, only female, preferably shirt/dress"*. The study is designed as within-subject to keep the number of testers at a reasonable size. Thus one group of people tests both variants. Which system is tested first is flipped in between subjects so that a bias because of learning effects could be reduced.

In order to create a realistic setup, it is necessary to generate a *dataset* that represents real-world items. For that purpose, we developed a dataset creation tool. The tool crawls clothing items from a well-known online clothing retailer website. To keep the amount of work reasonable, items were associated with an id, one of 19 types of clothing, one of 18 colors, one of 5 brands, the price (in Euro), the gender (male, female or unisex) and a list of links to the item's image. The resulting set is 2318 items strong, with 1141 for the male and 1177 for the female gender.

For the study we recruited *participants* of various age, educational background and current profession. Overall 30 people participated, whereas 33% of the users were female and 67% were male. The actual *testing procedure* used in the evaluation was structured as follows: We first asked the participants to provide background information about themselves, such as demographic information and their knowledge about mobile systems and recommender systems. Next, the idea of the system was introduced and the purpose of

the user study was made clear. We decided to choose a realistic scenario instead of asking users to find an item they like:

> **Task:** Imagine you want to buy new clothes for an event in a summer evening. You believe that the following types of clothes would be appropriate for this event: Shirt, t-shirt, polo shirt, dress, blouse or top. As colors you consider shades of blue, green, white, black and red. You have a budget of up to 100 €. You use the application to look for a product you might want to purchase.

After introducing the participants to the task, they received time to familiarize themselves with the user interface and grasp how the app works. After selecting and confirming the choice for a product, the task was completed. Then, testers were asked to rate statements about transparency, user control, efficiency and satisfaction based on their experience with the system on a 5-point Likert scale (from 1, strongly disagree to 5, strongly agree) and offer general feedback and observations. After having tested both variants, participants stated which variant they preferred and why. The full questionnaire can be found in *Appendix E*.

The testing framework applied in the user study is a sub-set of the aspects that are relevant for critiquing recommender systems and explanations. It follows the user-centric approach presented in [Pu et al., 2011a]. The measured data is divided into four areas: Transparency, user control, efficiency and satisfaction (see *section 2.3* for more details about the testing framework).

8.3.2 Results

The means of the measured values for the most important metrics of the two systems are shown in *table 8.3*. The baseline denotes the variant using only simple non-interactive explanations, EXP the version with interactive explanations. Next to the mean, the standard deviation is shown. The last column denotes the p-value of a one-tail paired t-test at a significance level of 0.05 with 29 degrees of freedom (30 participants - 1). Significant results are printed bold.

In order to measure actual understanding of the system's logic, users were asked to describe how the underlying recommendation system works after having tested each variant. In general, almost all of the participants could explain for both variants that the system builds a model of the user's preferences in each cycle and uses it to generate personalized recommendations. On average, when asked if a user understands the system's reasoning behind its recommendations, EXP performs better than the baseline (mean average of 4.63 compared to 4.3 out of a 5-point Likert scale). Further analysis suggests that the variant with interactive explanations (EXP) is perceived significantly more transparent than the variant with baseline explanations ($p = 0.018$). *Figure 8.6* illustrates the results.

	Baseline		EXP		p value
	mean	stdev	mean	stdev	
Perceived transparency	4.3	0.70	4.63	0.49	**.018**
Perceived control	3.23	1.04	4.33	0.71	**<.001**
Scrutability	3	1.31	4.36	0.85	**<.001**
Cycles	7.46	3.64	6.5	3.28	.14
Time consumption	160 s	74	165 s	83	.39
Perceived efficiency	3.43	1.13	4.33	0.75	**<.001**
Satisfaction	3.76	0.85	4.43	0.56	**<.001**

Table 8.3: The means of some important measured values comparing both variations of the system

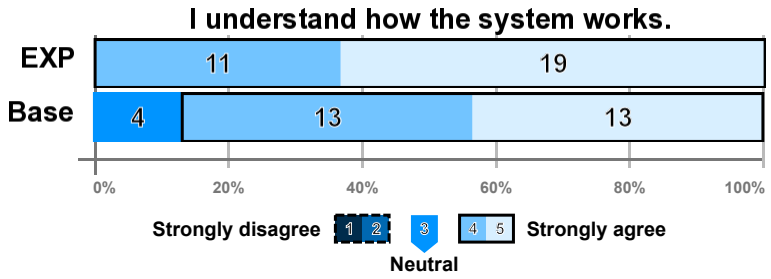


Figure 8.6: Distribution of ratings for perceived transparency on a 5-point Likert scale

Users were asked about the ease of telling the system what they are looking for in order to measure the overall user control they perceived. Average rating of participants was better with EXP (4.33 versus 3.23). In a further analysis, EXP seemed significantly better in terms of perceived overall control than the baseline ($p < 0.001$) which is represented in *Figure 8.7*.

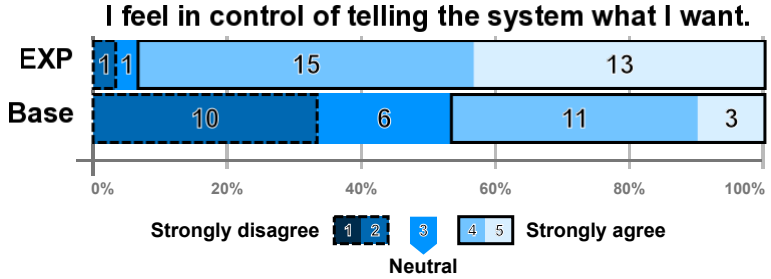


Figure 8.7: Distribution of ratings for perceived overall control on a 5-point Likert scale

When asked about the ease of correcting system mistakes, EXP performs much better than the baseline (mean average of 4.36 compared to 3 out of a 5-point Likert scale).

Further analysis reveals that EXP is significantly better in terms of perceived scrutability than the baseline ($p < 0.001$). See *figure 8.8* for an illustration.

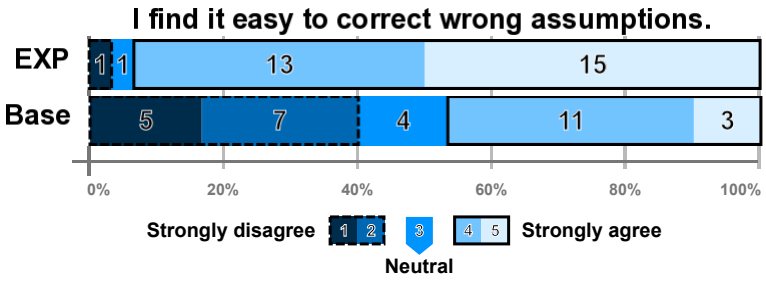
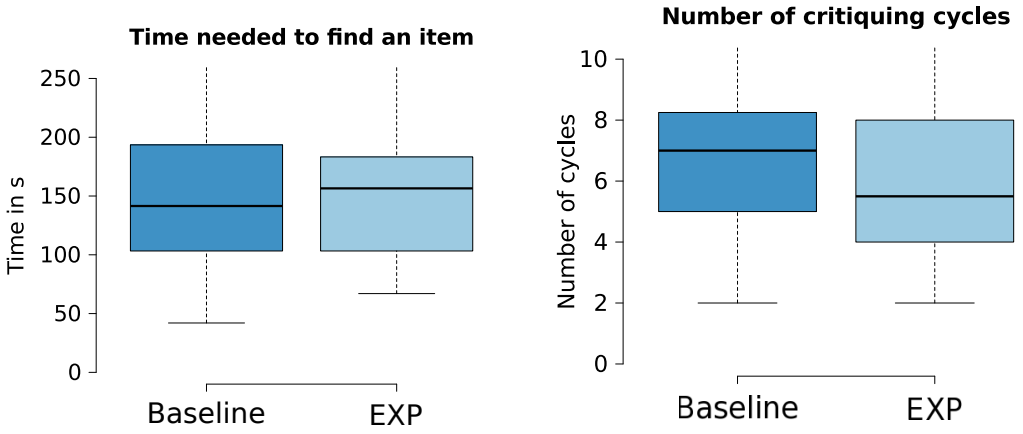


Figure 8.8: Distribution of ratings for perceived scrutability on a 5-point Likert scale

Participants completed their task on average one cycle less using EXP than the baseline (6.5 with EXP, 7.46 with the baseline). However, the one-tail t-test shows that EXP is not significantly better than the baseline ($p = 0.14$). The next part of measuring objective effort is done via tracking the time it took for each participant from seeing the initial set of recommendations until the target item was selected. On average, the baseline seems to be better with a mean session length of 160 seconds against 165 seconds. However, this observation is not significant ($p = 0.39$). One reason for this could be that although EXP gives its users tools to update preferences over several features quickly, it has more detailed explanations. Thus, users spent more time with reading. *Figure 8.9* shows a comparison of the time and cycles needed when completing the task with each variant.



(a) Required time to find an item

(b) Number of critiquing cycles to find an item

Figure 8.9: Box plots of the time in seconds (left) required to complete a session and number of critiquing cycles (the maximum has been omitted for space reasons)

We also asked the users about the ease of finding information and the effort required to use the system in order to measure the system's efficiency. The participants' average

rating was better with EXP (4.33 vs. 3.43 for the baseline). Further analysis revealed that users perceived EXP to be significantly more efficient than the baseline ($p < 0.001$). The result is illustrated in *figure 8.10*.

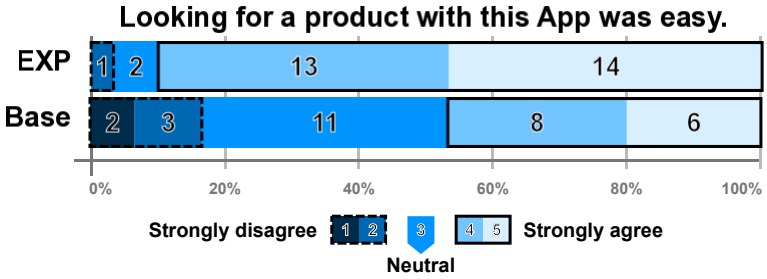


Figure 8.10: Distribution of ratings for perceived efficiency on a 5-point Likert scale

When inquired how satisfied participants were with the system overall, EXP performs better with 4.43 against 3.76. The one-tail t-test suggests that this is a significant result ($p < 0.001$). *Figure 8.11* shows a graphical representation.

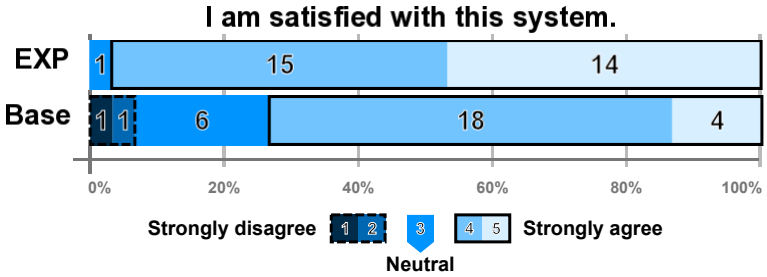


Figure 8.11: Distribution of ratings for satisfaction on a 5-point Likert scale

Finally, we asked the participants to pick the favored variant. 90% preferred the variant with interactive explanations (EXP) over the variant with simple non-interactive explanations (baseline), see *figure 8.12*. In general, participants perceived more control over recommendations and stating their preferences with EXP. 80% of participants mentioned it as one of the reasons to select EXP. Moreover, interactive explanations became another deciding factor for more than half of the participants who chose EXP. All of those few participants who preferred the baseline noted that they found it simpler to use.

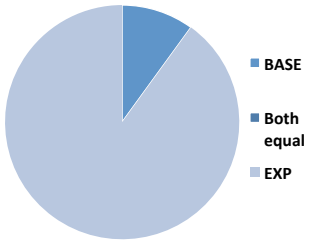


Figure 8.12: Preferred variant

Within this user study, we were able to show that our proposed concept of generating mobile interactive explanations performed significantly better compared to the approach with non-interactive simple explanations in terms of our main goal to increase transparency and scrutability. Also our side goals to improve perceived efficiency and satisfaction could

be reached. The users generally liked the user interface and appreciated the control over the system due to the interactive explanations. Eventually, the proposed concept led to a highly accepted recommender system and more participants preferred using the system offering interactive explanations compared to the baseline. Results of the user study also demonstrate the user appreciation of transparency and control over the recommendation process in a conversation-based Active Learning mobile recommender system.

8.4 Conclusion and Next Steps

This chapter investigated the development and impact of a concept featuring interactive explanations for Active Learning critique-based mobile recommender systems in the fashion domain. The developed concept proposes the generation of interactive explanations to make the system more transparent while also using them as an enabler for user control in the recommendation process. A method is developed to generate explanations based on a content-based recommendation approach. Due to the interactivity the user gets the chance to correct possible system mistakes. In order to measure the applicability of the concept, a mobile application using the proposed explanation generation algorithm was developed and evaluated. The proposed concept performed significantly better compared to the approach with non-interactive simple explanations in terms of our main goals to increase transparency and scrutability and side goals to increase perceived efficiency and satisfaction. This chapter concludes the investigation of our fifth and last building block of our conceptual framework presented in *chapter 3*. We will now evaluate the extended context-aware recommendation approach, as well as the results regarding the other building blocks in an overall user study to find out, if the application of our conceptual framework improves the overall user experience of mobile recommender systems.

9

Evaluation of the Final Prototype

This chapter focuses on the final evaluation of our conceptual framework. We develop a prototype that implements all investigated five building blocks and evaluate if a positive user experience is achieved in a user study with 100 participants. Although each element has already been evaluated separately, we now want to test the interaction of the components and some optimizations of the previous approaches that have not been tested yet in order to examine if the positive user experience of the mobile recommender system is maintained . The first section (1) defines the prototype requirements. We also point out the characteristics and the interaction design of the developed prototype and highlight its distinctions to the baseline. Next, the goals of the overall evaluation and the undertaken measurements are presented. We also describe the setup of the user study, how we conducted it and its results (2). The final section (3) concludes this chapter by discussing the results of the study.

9.1 Final Prototype

The individual building blocks of our proposed conceptual framework of *section 3* have already been investigated and evaluated separately with promising results. Since the design science methodology provides for an evaluation of the created artifacts, we now want to study if the interaction of all these building blocks still creates a positive user experience when combined in one system. Moreover, we undertake some optimizations of the proposed approaches (in particular regarding the context-awareness, *chapter 7* and the generation of explanations, *chapter 8*) that have not been evaluated yet. We therefore develop a final prototype that combines all of these approaches we came up with when investigating each building block.

9.1.1 Prototype Requirements

The development of the final prototype aims at evaluating if the interaction of the concepts developed for each building blocks still creates a positive user experience when combined in one application. The prototype has to fulfill several requirements in order to allow an accurate evaluation. Biases in favor of the new prototype shall be avoided by comparing it with a baseline system. Both variants should look similar in order to not influence users by the fact that one system was obviously developed more sophisticated and uses more information than the other system. Besides this, the questionnaires should be designed such that no inferences on the more sophisticated system can be drawn and that the different recommendation approaches of the baseline and the final prototype can be compared. Five key features of the previously investigated framework elements which have proven successful should now also be applied for the new prototype. These features are in particular:

Requirement 1: For the calculation of the recommendations, the algorithm presented in *chapter 4* that is based on *Active Learning* and allows for critiquing should be used. Also the diversity-based approach which has proven to generate better results than the similarity-based approach should be maintained, especially in the beginning of the recommendation session, where the user might look for inspiration.

Requirement 2: As *user model*, the stereotype-based approach that has been developed in *chapter 5* and was able to decrease the cold-start problem when the mobile recommendation system was used for the first time, should be taken over.

Requirement 3: Different *interaction designs* for a mobile recommender system have been evaluated in *chapter 6*. The derived guidelines that allow an intuitive interaction with the system should be considered for the interaction design of the final prototype.

Requirement 4: The mobile *context* has been taken into account for the generation of recommendations in *chapter 7*. The contextual factors that have been identified to enhance the recommendations should also be integrated in the algorithm of the new prototype.

Requirement 5: In *chapter 8* we came up with a solution of how to automatically generate interactive mobile *explanations* of the recommendations. Results showed that this feature improved the user experience and should therefore also be implemented in the sophisticated prototype.

The final prototype should be developed for an exploratory shopping scenario where the user has no specific item in mind and is looking for an inspiration. The approaches developed in the previous chapters have all been tested in a mobile shopping environment. To achieve better comparability to the previous results, we again choose a mobile shopping recommender system as application scenario.

9.2 Prototype Implementation

In order to test if the interaction of all building blocks still creates a positive user experience when combined in one system, we develop a prototype that implements all previously developed features, that have already been evaluated individually. For our application we imagine a user that favors shopping locally instead of shopping online. However, she prefers being inspired by a recommender system before visiting a specific shop, where clothing items she likes might be available. Although the application does not save any information about the user, it keeps track of all the user's critiques and incorporates all these critiques into the recommendation of the next items. The following sections describe our dataset and specify which features of the previously proposed approaches have been taken over and which changes have been made.

9.2.1 The Dataset

For the previously developed applications we already created several sets of clothing items. However, this dataset was quite outdated as many of the items were taken offline, and therefore most of the pictures could not be accessed anymore. As the fashion trends also change over time, we decided to generate a larger dataset for the final prototype. With the now deprecated *Google Search API for Shopping*, the approach for generating such a dataset could no longer be used (see *section 4.3.1*). A promising alternative is the free shopping *API* of the online shopping retailer *Zalando*, where all relevant data for items can be accessed [Zalando, 2015]. In order to reach a good variety of items for both genders, we selected 16 categories of clothing and added the most popular 200 items out of each category to the dataset. Unfortunately, some discrepancies between the pictures and the colors caused problems in pre-tests, so the whole dataset had to be corrected manually. The final dataset, created in March 2015, contains 5157 items. Out of these items 2773 items are for women, 2262 for men and 122 items are labeled as unisex. The items are from 450 different brands, are assigned to 17 different clothing types and differentiated into 18 colors.

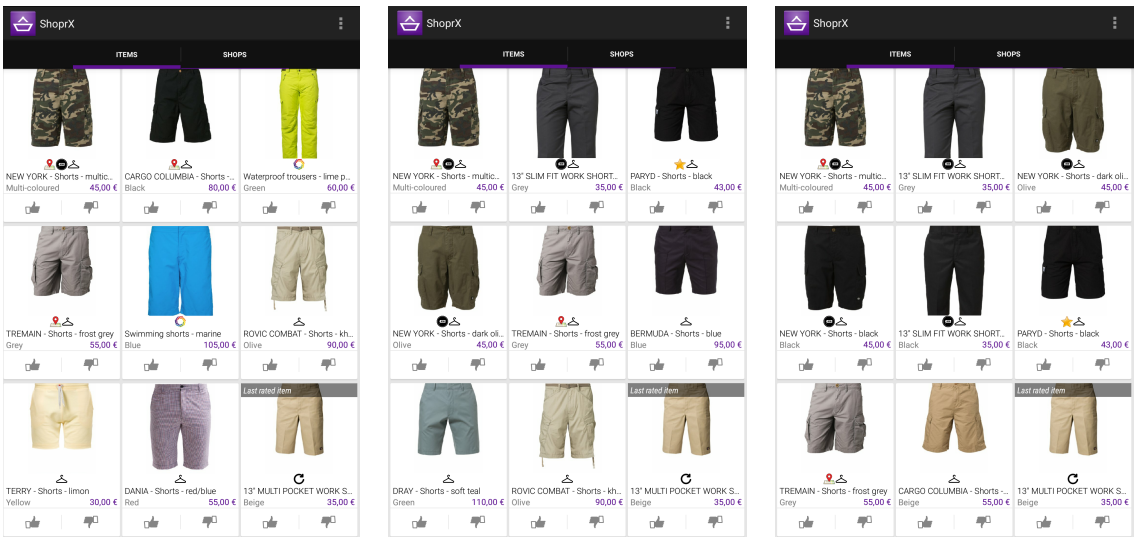
9.2.2 Integration of Active Learning

The final prototype also uses Active Learning to personalize the recommendations. In order to inspire the user at the beginning of the recommendation process a diverse set of items is shown. The user can then criticize one of the items by liking or disliking it. Subsequently she criticizes specific features of the item (e.g., color or price). Based on this critique a new set of recommendations is presented, which can be criticized again. In case the user got stuck, it is possible to restart the recommendation process, e.g., to focus on a different type of items. Compared to the Active Learning approach developed in *chapter 4* some adaptations to the content-based recommender have been necessary. As the attribute *brand* was now included in the dataset, we created a new similarity graph for brand and

adapted the similarity graphs for color and clothing type to adequately represent the new dataset. The similarity graph for brands was created using the stereotypes integrated into the system (see *section 9.2.3*). As all stereotypes define which brands represent the particular stereotype the most, an algorithm which compares these assigned brands was created. The more often two brands share the same rating within a stereotype, the more similar they are. For example if *Ralph Lauren* and *Lacoste* shared the same value for at least five of the eight defined stereotypes they are considered to be similar. For more details on similarity graphs refer to *chapter 4*.

The dataset for the evaluation of this prototype is much larger than in the previous applications. Although we introduced more categories (e.g., for clothing type), the content-based recommender system showed items that were too similar to the current user's preferences. If the user liked "shirt" and "white" she was only presented with white shirts, hence there was no diversity in the recommendations. It was also possible to get stuck within the recommendations, such that whatever (positive) critique is provided, the system did not show new items. To solve this problem, [Alodhaibi et al., 2011] propose using a randomized greedy nearest neighbor strategy such as [McGinty and Smyth, 2003a] do. To integrate diversity into positive critiques in this system we also decided to use the bounded greedy selection algorithm by [McGinty and Smyth, 2003a], as is already done on negative critiques. Now, it is also used for positive critiques, with an α of 0.997, which allows some randomness, but still mainly focuses on the similarity of items. For an overview on the effect of different values for α see the screenshots in *figure 9.1*. The shown pictures are representative for other preference selections as well. Below an α of 0.95 there were no changes to the recommended items. For a value of 0.997 the algorithm showed a diverse set of items, which was mainly dominated by the elicited preferences, but also influenced by the diversity through the bounded greedy selection algorithm. Regardless of this, there has still been the problem that the recommendations did not change significantly when new preferences were elicited, as a lot of the former shown items still fulfilled (parts of) the query. Therefore, a method was developed, which ensures that items that were shown to the user are not shown within the next three critiquing cycles. After these three cycles, it is possible that the items are shown again if the user's preferences did not change significantly although it is expected, that the user is now searching in a different part of the search space. Excluded from this mechanism is the item that the user selected to criticize, as she might want to further criticize it to improve recommendations. This item is always shown as last of all nine items.

One problem with the already implemented bounded greedy selection algorithm is that it takes about eight seconds to calculate the quality (the distance metric for diversity) for about 1000 products and to select the top 20 products on a Samsung Galaxy S3 mini. We expect that the user is not willing to wait that long until recommendations are displayed and that this waiting time significantly negatively affects the user experience. Therefore, the algorithm had to be accelerated. In the algorithm described in *chapter 4* the number of similarity calculations increases by n in each round of selecting the next recommendation. This makes the effort for running the algorithm $\mathcal{O}(n^2)$. However, this algorithm can also run in linear time by adapting the algorithm for the determination of the quality. The



(a) $\alpha = 0.95$ (b) $\alpha = 0.997$ (c) $\alpha = 1$

Figure 9.1: Selection of type 'shorts' and brand 'Dickies' for different α levels

algorithm calculates the quality of all items to recommend (*Recommendations*) to the case base (*CaseBaseAdapted*). Hence, the algorithm performs redundant work, as only the quality to the latest recommended item (r) has to be calculated and all other quality-values can be cached (see *line 25* of *algorithm 4* where the *relDiv* value of the current item is always added up and not calculated again for all the other items as has been done in the previous approach). The resulting algorithm can be seen in *algorithm 4*. It finishes within 0.6 seconds for the same amount of items and has a complexity of $\mathcal{O}(n)$. Again, C refers to the case base which contains all the items c, t denotes the target query, k the amount of items that are recommended to the user (we again use a 3 by 3 grid view and one spot is reserved for the recently criticized item) and b refers to the *bound* ($b = 10$).

9.2.3 Integration of Stereotypes

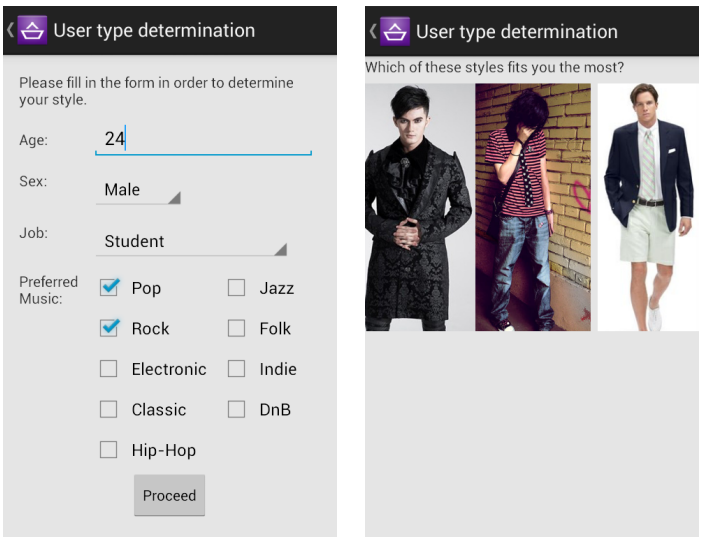
In *chapter 5* it was argued that content-based recommender systems face the new user problem. Moreover it was shown that the integration of stereotypes into a critique-based mobile shopping recommender system can improve the recommendation quality. The stereotypes were therefore defined as if they have rated specific features of items. These *ratings* are used as initial preference profile for our final prototype, as well as for the baseline. Stereotypes can function as a full user profile without the necessity to rate lots of items. Nevertheless, they are insufficient generalizations of individual preferences. Hence, the more preferences a user explicitly states via critiquing items, the less accurate it is to use the stereotype's preference profile. By combining the stereotype's and user's preferences, ratings for all items are available (via the stereotype), whereas they become less relevant the more items a user criticizes. At the beginning of the usage of the application, the stereotype based

Algorithm 4 Adapted bounded greedy selection

```
1: procedure BoundedGreedySelection(t, C, k, b)
2:     //Select the bn most similar items to query
3:     CaseBaseAdapted ← mostSimilarItemsToQuery(C, b · k)
4:     Recommendations ← {}
5:     //Initialize the last recommended item
6:     r ← null
7:     for j = 1 to k do
8:         for all c ∈ CaseBaseAdapted do
9:             c_quality ← QualityOfItem(t, c, r, j)
10:        end for
11:        sortCaseBaseByItemQuality(CaseBaseAdapted)
12:        r ← first(CaseBaseAdapted)
13:        Recommendations ← Recommendations + r
14:        CaseBaseAdapted ← CaseBaseAdapted − first(CaseBaseAdapted)
15:    end for
16:    return recommendations
17: end procedure
18:
19: //Where...
20: α = 0.997
21: QualityOfItem(t, c, r, j) = α · Similarity(t, c) + (1 − α) · RelDiv(c, r, j)
22:
23: procedure RelDiv(c, r, j)
24:    if r ≠ null then
25:        c_relDiv ← 1 − Similarity(c, r) + c_relDiv
26:        return c_relDiv/j
27:    else
28:        return 1
29:    end if
30: end procedure
```

recommender system presents a catalog of questions to the user. The user has to state her age, gender, job and music taste (see *figure 9.2a*). We use these inputs to calculate three stereotypes, that could describe the user's clothing style best. Pictures of these stereotypes' clothing styles are subsequently presented to the user (see *figure 9.2b*), who selects the most adequate one. The possible stereotypes are *Athlete/Jock, Classy, Emo, Girly, Gothic, Indie/Hipster, Mainstream, Preppy* and *Skater* (*Urban* has been removed in the final prototype, as it could not be adequately distinguished from Mainstream). For a more detailed overview on the classification of these stereotypes and the rationale behind this see *chapter 5*.

As mentioned in [Shardanand and Maes, 1995] the assignment of users to stereotypes is a collaborative filtering approach. Thus, the system becomes a hybrid recommender.



(a) Stereotype questionnaire (b) Top three stereotypes

Figure 9.2: Determine the user's stereotype

Adomavicius and Tuzhilin propose four approaches to combine a content-based recommender system with collaborative filtering. The four approaches are first, calculating the algorithms on their own and combining them, second, enhancing the content-based recommender system by characteristics of collaborative filtering, third, enhancing collaborative filtering with content-based characteristics and fourth, combining both methods in a unifying model [Adomavicius and Tuzhilin, 2005]. We choose the first approach due to the following reasons:

1. The stereotype algorithm has to be executed only once since the results can be reused afterwards, as the user's stereotype does not change. This is comparatively more time efficient than combining both algorithms into a single one or integrating collaborative filtering into the content-based recommender system. Therefore, the second and fourth approaches are insufficient.

2. As the critique-based recommender algorithm is the *main* algorithm, the content-based recommendations should not be incorporated into the collaborative filtering approach. This makes the third approach insufficient.

3. As the user's preferences, elicited by the critique-based recommender system, become more specific during the recommendation process, the importance of the stereotype-based recommender system decreases. This is easily and sufficiently possible to implement with a weighting scheme.

To combine these two algorithms, they are first computed on their own. Then, the items are ordered by their similarity to the current preferences (content-based). If two items

equally match the preferences (based on the score for the content-based recommendation), the score of the stereotype's rating is compared. This has the following effects:

1. At the beginning of the recommendation process, the user has not entered any preferences yet. Therefore, all items will be equally similar to the user's preferences. However, they are ordered by their similarity to the stereotype.

2. With each preference the user explicitly states (each critique), the influence of the stereotype-based recommender decreases and the items can be distinguished based on their similarity to the current preferences. Each stated preference groups the items, which are equally similar to the user's preferences. Hence, the stereotype preferences only have to be applied to order these sub-groups.
   For example a user likes the color *green* in her first critique. If there were 100 items, out of which 15 are green, these green items are ranked higher than all other items. Furthermore, similar colors like olive will get a higher similarity to the user's preference. Within these groups (green items, olive items, other items) they will be sorted according to their stereotype based ranking and the items ranked at the top are presented to the user. If the user specified a preference for trousers in the next step, the influence of the stereotype based recommender further decreases as there are not many green (or olive) trousers.

3. In the end the stereotype-based recommender system will have contributed to the effective elicitation of the user's preferences by providing the best training points according to the user's interests. As the recommendations are based on the explicit preferences, the influence of the stereotype will get lower and lower because the real preferences of the user are known.

All in all the integration of stereotypes is expected to improve the recommendation quality especially at the beginning. The algorithm for stereotype-based filtering is losing predictive power during the recommendation session, which is expected and desirable.

9.2.4 Integration of Context-Awareness

The results of the user study (*subsection 7.3*) regarding the integration of mobile context into the recommendation process were very promising. However, we decided to undertake some adjustments of the context-aware recommendation algorithm for the final prototype. The following concept of a context-aware mobile recommender system has also been published in [Lamche et al., 2015a].

We imagine a system that uses the user's mobile context to recommend clothing items available in shops close to the user's position and also allows critiquing. As described in *chapter 7 (section 7.1.1)*, context can be integrated into the recommender system in three different ways: Contextual pre-filtering, contextual post-filtering and contextual modeling. We will combine two approaches (contextual pre-filtering and contextual post-filtering) to

improve the recommendations (see *figure 9.3*). Pre-filtering is used to determine which items of the case base are relevant to the user. Relevancy for example depends on the distance the user accepts to travel, or the opening hours of a shop. Post-filtering is used to filter the items that shall be recommended according to their adequacy to the current context by using a nearest neighbor algorithm. In order to build a database of contextually tagged items, a pre-study was executed asking users to classify items according to contexts. This data ensures, that some items are already contextually tagged, which is needed for the post-filtering of the recommendations.

Contextual Pre-Filtering

In the contextual pre-filtering step, we make sure that only relevant data is loaded into the recommender system. Therefore, the context factors *distance to shop, shop crowdedness, shop opening hours* and *item in stock* are used to restrict the case base and avoid unnecessary search in items the user does not want to see. The case base is filtered in four steps. First, all shops that are not within the specified distance, then shops that are not open at the specified time and shops that do not match the crowdedness criterion are excluded. Finally, it is verified that the item is in stock. After pre-filtering the items based on these conditions, it is verified that at least 300 items are available in the case base, as our tests showed that this is the minimum amount of data to adequately react to the user's preferences. However, if there were not enough items available in the case base, these conditions are relaxed and the user is notified about this step. Out of these 300 items, the content-based recommender algorithm selects 20 items based on the user's stated preferences, which are then the input of the contextual post-filtering algorithm.

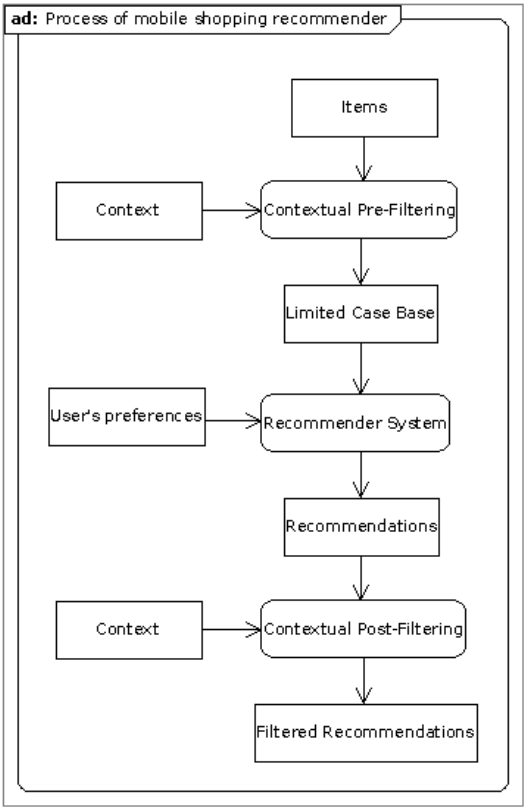


Figure 9.3: The context-aware shopping recommender process

Acquisition of Context Relevance

Before being able to recommend items based on context, the relevant context has to be defined. We assess the following context factors as relevant for our context-aware mobile shopping recommender system: *Time of the day, day of the week, temperature, weather, company, distance to shop, crowdedness, shop opening hours* and *item is in stock.* The

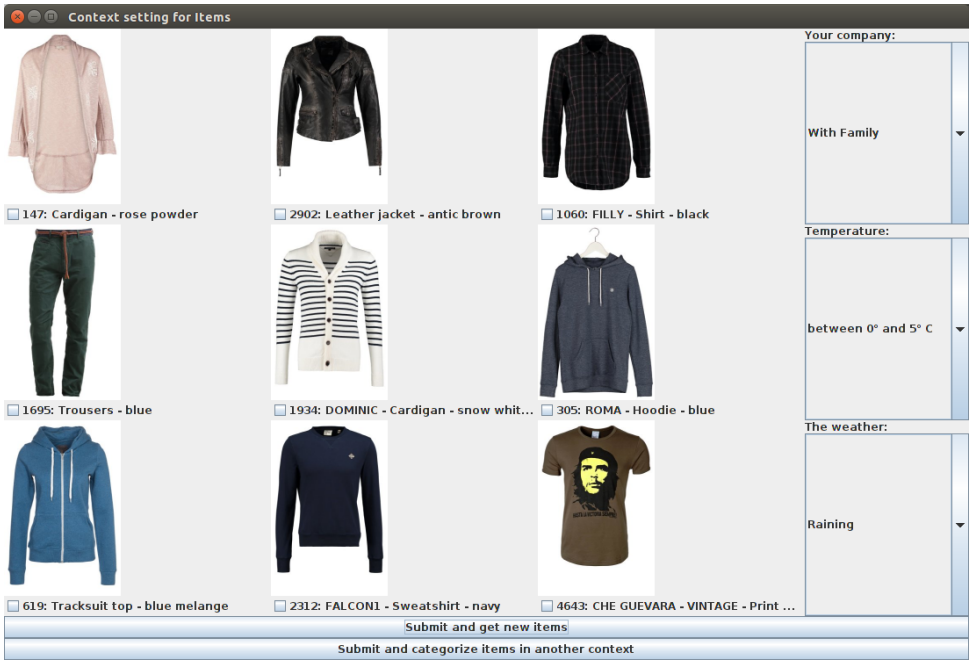


Figure 9.4: Tool for elicitation of item preferences in contexts

reasons for the selection of the context factors are described in detail in *Appendix F.1.* In order to acquire contextual ratings, a convenience sample of the target population was asked to specify which items they are likely to buy in a specified context, similar to the approach specified in *section 7.2.1.* We developed a simple Java tool (*figure 9.4*) which shows nine pictures and descriptions of clothing items. The testers could specify if they would consider buying the product depending on a randomly selected company, temperature or weather, which is specified on the right side of the tool. Overall 747 contextual ratings for 674 different items were created by 6 users. This data forms the basis for the decision generation in the contextual post-filtering algorithm.

Contextual Post-Filtering

Out of 20 calculated items in the pre-filtering process, only nine items are actually displayed. Therefore, the contextual post-filtering algorithm (illustrated in *algorithm 5*) has to eliminate eleven items in each cycle. The context factors time of the day, day of the week, company, temperature and weather are used to post-filter the recommendations. For this purpose, we use a k-nearest neighbor method because this technique has proven to be adequate in different context-aware recommender systems, especially [Panniello and Gorgoglione, 2011], [Lee and Kwon, 2013], [Dao et al., 2012]. The most important component in nearest neighbor algorithms is the used distance metric. In our approach, the user is not able to rate an item within a given context, but only to select it (and therefore implicitly rating it as good). Based on this consideration, we came up with a distance metric that

defines an *average context* in which an item is selected. The average context specifies in which context an item is selected. If an item was not selected in any context, it can be assumed, that this item neither is liked by a lot of users nor in a specific context and can therefore receive a higher distance to the current context. Popular items, which are selected in many different contexts will receive a distance which is close to 0.5. However, as they are very popular, they should not receive a high distance and therefore their distance is reduced by a defined percentage of their distance.

$$avgContextDist(c,i) = \frac{\sum\limits_{t \in i_c} w_{i,t} \cdot dist(c_f, t)}{N(i_t)} \cdot \frac{N(c_f)}{\sum\limits_{f \in i_f} w_f} \tag{9.1}$$

Equation 9.1 defines the distance metric. It calculates the distance between an item's (i) average context (in which the item is selected) and the current context (c). The first quotient calculates the average distance to the current context. Therefore, the distance of each context condition to the current context is calculated and summed up. The set of all context conditions in which an item has been chosen is defined by i_c. An individual context condition in which an item has been chosen is defined by t. For each clothing type, the context factors are of different importance. Hence, different weights ($w_{i,t}$) can be assigned to context conditions. We assigned the weights for each clothing type based on the results of the experiment conducted in *chapter 7 (section 7.2.3)*. The distance function $dist(c_f, t)$ (*equation 9.2*) calculates the distance between the current context condition c_f and the context condition t in which the item was chosen. The number of context conditions in which an item has been chosen is defined by $N(i_t)$ (in this work $N(i_t)$ always is a multiple of five - the number of context factors). In order to make different items (with different overall weights) comparable, they are scaled between zero and one by multiplying with the second quotient of the function. The context factor weights have already been calculated in *section 7.2.1*, however we now extended the case base to include seven new types of clothes. The weights of the new context factors are listed in *Appendix F.2*. $N(c_f)$ defines the number of context factors available. This is divided by the sum of weights of all context factors (w_f) for this item ($f \in i_f$).

$$dist(c,t) = \begin{cases} graphDistance(c,t) & \text{if } t \text{ is nominal} \\ \frac{|c-t|}{range_t} & \text{otherwise} \end{cases} \tag{9.2}$$

If the context factor is ordinal, interval or ratio-scaled, the distances are calculated based on the euclidean distance. Otherwise the *graphDistance*, a pre-defined distance for nominal attributes, is used. This *graphDistance* is similar to the distance used by Lee and Lee [Lee and Lee, 2007]. The context factors weather and company use this *graphDistance* and define an undirected graph with distances between all context conditions (e.g., the weather conditions *Sunny* and *Rainy* have a higher distance than *Sunny* and *Cloudy*). The assigned distances are used as an input for the distance method and can be seen in *Appendix F.3*. For the context factor time of the day we use a cycle, as the afternoon ends with the night, whereas the night is the first part of the day. For all other conditions it is expected that the euclidean distance provides good results. Although we want to achieve a high item frequency, we consider very popular items as being interesting for the user, especially in

a shopping scenario. Therefore we alter the resulting distance ($avgContextDist(c, i)$) if the item was selected in more than 30% of all contexts: The item's distance is reduced by 20 % so that it is more likely to be displayed to the user. Every item that was not been selected in any context receives a distance of 0.51. We came up with this value because it is the average distance at the second tertile when considering all distances of items rated in a specific context to a randomly selected context. This ensures that items which have not been rated within a specific context in our pre-study (see *subsection 9.2.4*) are more likely to be presented to the user than items that were considered as being uninteresting in that specific context. The whole algorithm for contextual post-filtering is as *algorithm 5*.

Algorithm 5 Post-filtering by current and item context

```
1:  procedure ContextualPostFiltering(items, currentContext, n)
2:      cContext ← currentContext
3:      for all item in items do
4:          itemContexts ← item.getSelectedInContexts()
5:          overallItemDistance ← 0
6:          overallContextFactorsForItem ← 0
7:          for all iContext in itemContexts do
8:              overallContextFactorsForItem ← overallContextFactorsForItem + 1
9:              factorWeight ← iContext.getWeight()
10:             //Calculate distance between current context and item context
11:             itemDistance ← itemDistance + distance(cContext, iContext) · weight
12:         end for
13:         //Divide distance by number of context factors
14:         itemDistance ← itemDistance/overallContextFactorsForItem
15:         //numberOfContextFactors: How many context factors did the user set? For
16:         //this prototype always five (time of the day, day of the week, temperature,
17:         //weather, company)
18:         //Scale between 0 and 1
19:         itemDistance ← itemDistance·
20:         (numberOfContextFactors/maximumContextWeightsForItem)
21:     end for
22:     setBonusForPopularItems(items)
23:     setDistanceForNonSelectedItems(items)
24:     //Selects the n closest items to the current context
25:     recommendations ← selectClosestItems(items, n)
26:     return recommendations
27: end procedure
```

The algorithm's disadvantage is that it weights each factor independently without taking into consideration possible connections between the individual context factors. For example the connection of rain and being with a friend might be more different from rain and being with the family, than the individual distances between being with the fam-

ily and being with a friend. This detection of dependencies could be done by decision trees or other machine learning techniques. Nevertheless, it is expected that the algorithm provides reasonable recommendations for the user's current context without these dependencies. The recommendation algorithm calculates the context distances in less than 100 ms on a Samsung Galaxy S3 mini for 20 items with the items being set in (overall) 200 different contexts. It allows weighting of context factors for each clothing type separately and distances for nominal attributes. The method $selectClosestItems(items, n)$ sorts the items by their distance to the current context. In case of any ties (meaning that two items have the same context distance) the system uses the similarity measure to calculate how similar the item is to the user's preferences and has already been applied in *chapter 4*.

9.2.5 Integration of Explanations

Based on the promising results of the user study that compared a mobile recommender system with interactive explanations to a system with a very simple explanation approach (see *chapter 8*), we want to investigate some more aspects of this research area and integrate them in the final prototype. In particular, we came up with the idea of integrating expert knowledge, as well as context information into our explanation generation approach, in order to increase persuasiveness and build trust in the system, an approach that has not been investigated in previous researches as can be seen in *chapter 8 (subsection 8.1.2)*.

Considering Context

The key to generating useful explanations lies in the information used to calculate the recommendations. The explanations generation method described in *chapter 8 (subsection 8.2.2)* can only be used for the content features of the items and not for context information. We want to take two types of context information into account. The first one is the location of the user and the surrounding shops. It is used to increase persuasiveness of recommendations, when a shop is nearby. The second one is combined of multiple context factors concerning the user and her surroundings. It helps to filter the recommendations before showing them to the user. The location context was already considered in our previous approach and will therefore be reused. *Explanation Score* for the location is selected using domain knowledge. The closer a shop is, the higher the score. We slightly adapted the original formula calculating the *Explanation Score*: $ES_{I,D} = LS_{I,D} \cdot \omega_D$. This approach might lead to justifying recommendations mainly with the dimension that has the lowest amount of feature values and in particular if a database with a high amount of brands will be used, the recommendation algorithm will almost ignore the label. Therefore, the recommendation as well as the explanation algorithm was modified, so that every dimension is weighed equally. This can simply be achieved by multiplying the *Local Score* with the range of the dimension. The resulting formula is the following:

$$ES_{I,D} = length_{total\_attribute\_values} \cdot \sum_{i=0}^{n-1} \omega_{I,D}(i) \cdot \omega_{Q,D}(i) \tag{9.3}$$

Considering the *Information Score*, the original formula $IS_D = \frac{R+I}{2}$ was used. While R was calculated identically, I needs some slight adjustments. $I = \frac{n-h}{n-1}$ stays the same, but instead of values with the same characteristic for a dimension, h represents the item's similar *Explanation Scores* here. To do so, they were compared to a defined threshold. Without other dimensions, it is not possible to calculate a *Global Score*.

The context factors used for filtering were defined in the previous *subsection 9.2.4*. The system takes five different factors into account: Weather, temperature, company (e.g., being alone or with friends), time of the day and day of the week. While they might all influence the customer, only weather and temperature are suitable for explanations, as the other aspects work more on a subconscious level. Revealing the user that the system expects her to buy more conservative clothing because she is accompanied by her mother might even have a negative effect. In addition, it is difficult to explain these context factors to an average user who is not trained in statistics or psychology. Weather and temperature however, are logical arguments that people can ignore, but will accept as understandable reasoning. For this reasons, our system will only consider information about those two context factors for the generation of explanations. The data about context information had bigger variations considering feature values. For example, defining *Range* and *Information* is complicated for the context factors weather and temperature. Thus, the data was hard to map on *Explanation-* and *Information Score*. Also, missing values for some items were a big problem. Thus, it was not possible to implement the scores in this approach and we therefore introduced two individual thresholds for the dimensions in order to provide context explanations. One being a fix context distance value, representing the maximum distance that still makes a recommendation possible. The other one being an average performance of the dimension. For example, an explanation using weather should be prevented when the score for the other context factors was much better. After performing some experiments, we determined values for the thresholds that delivered reasonable results.

Considering Expert Knowledge

Expert knowledge is great for recommender systems, as it can be used to make high quality recommendations independent of the preference of a user. For a shopping recommender in the area of fashion, the best expert knowledge is about the newest fashion trends. A solution to get real expert knowledge on products from *Zalando* (the online fashion retailer we used to create our dataset, see *subsection 9.2.1*) was found on the website *Stylight*[1], where fashion bloggers post combinations of outfits they like. Among those bloggers are several who were nominated for the *Influence Awards*[2], an award ceremony hosted by *Stylight*. Each style outfit is also linked to similar products in different online shops, including *Zalando*. Those items can then be added to the dataset, or marked as *trendy* in case they were already in it. If one of these items is suggested to the user, a simple explanation informs the user, that this item is *"suggested by our fashion experts"*.

[1] `http://www.stylight.com/`

[2] `http://influencerawards.stylight.com/`

Figure 9.5: Icons for color, price, brand, clothing type, weather, temperature, location, expert knowledge, diversity and last critique

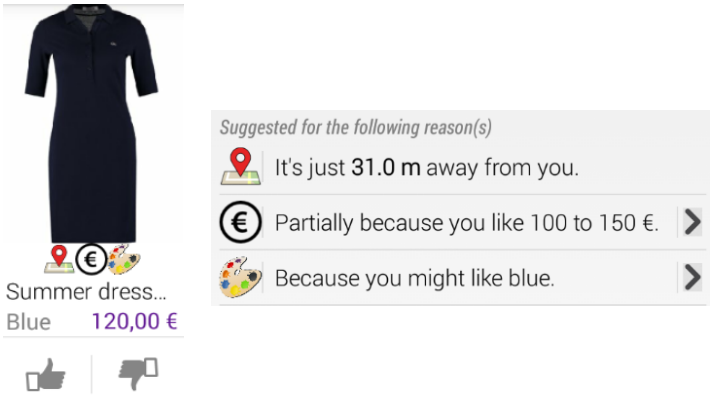
When clicking on the explanation, a small pop-up window opens with a more detailed description. The text also aims to inform the user how the item was selected as trendy. It is important that the user knows it is not just more expensive or advertised, but was selected by independent fashion bloggers.

Classifying Arguments

After calculating the different scores, the next step is to decide which arguments for the explanations to choose and how to use them. Apart from the already described arguments in *chapter 8 (subsection 8.2.2)*, we now introduce two new argument types: *Context-based arguments* and *expert knowledge*. Explaining an item with a partially fitting context is not a good approach, especially when considering that context information is often not 100% accurate. Suggesting swimwear when it is between 15 and 20 degrees is a good example why this would be problematic. The explanation "it might get warmer" is not very convincing, considering how reliable weather forecasts tend to be. That is why *context-based arguments* need to exceed a specific threshold like *strong primary arguments* (see also *figure 8.1* of *chapter 8*). Last but not least, *expert knowledge* will be used to form arguments. How this argument type looks like and is generated has already been described in the previous subsection.

Content Selection

In total, an item can have eight different explanations at the same time. Four for the different item attributes, two for the weather context, one for the location context and another one for expert knowledge. Showing all of them at once is not a good idea, because they would fill the whole screen and the user would be overloaded with information. We limit the number of explanations to a maximum of five per item. Expert knowledge has the highest priority and is always shown first. The reason is that it is not obvious for the user if a suggestion is based on expert knowledge. Next comes the location context, because the distance is also relevant for the user's choice and is not obvious otherwise. The other context arguments such as weather are also important but can more easily be predicted by the user. They just justify for example why a jacket is higher suggested than a dress in winter, even if it fits the customer's preferences less. The explanations describing the user's clothing preference are shown last because they have been stated explicitly during the recommendation process.



(a) Basic explanations

(b) Detailed explanations

Figure 9.6: Visualization of explanations

Visualizing Explanations

Our system now considers context information and expert knowledge, so text-based explanations would get very long. A possible solution to the problem is the usage of icons. They can be much smaller than a text explanation and, after initially learning what they stand for, a user will recognize them and remember their meaning. We therefore want to investigate the usage of icons by giving every dimension its own icon. Using icons with a short associated text also works great together with an argument styled presentation of explanations as described by Briguez et al. [Briguez et al., 2014]. The items were chosen carefully, so a user can easily associate the correct explanation. *figure 9.5* shows the icons used for the explanations. Considering a natural learning process, the degree of detail of an explanation should vary. We suggest three layers of explanation details. The most basic layer is part of the item list on the main screen and only shows icons without any text (see *figure 9.6 a*). The second layer is reached by either selecting or rating an item and contains explanations with icons and text (see *figure 9.6 b*). A third, final layer is reached by clicking on the second layer explanations. This layer is a detailed explanation concerning all information in the context of that argument. To hint that clicking is possible, a small arrow on the right side is added.

Correcting Wrong Assumptions

We also made some improvements regarding the correction of wrong assumptions of the system. In the new approach, the user is now allowed to mark the different values she likes or dislikes, instead of just selecting those she likes (see *figure 9.7*). Clicking once on an item value (e.g., on the brand 'Adidas') marks the value as preferred, clicking twice on a value (e.g., on the brand 'Escada') marks it as disliked. Although it might seem like a minor change it has a big effect. For example if we image that a user selected green and

blue as her favorite colors, the resulting query in the previous approach looks like this:

$$q = \{color_{red,blue,green,white,black}(0, 0.5, 0.5, 0, 0)\}$$

All but the selected values are set to zero, so further browsing is hardly possible. To get back to other colors, the system has to be corrected again. Now, the same input results in another query:

$$q = \{color_{red,blue,green,white,black}(0.14, 0.28, 0.28, 0.14, 0.14)\}$$

A liked value gets a doubled score compared to the others so it is clearly preferred by the algorithm. Still the system allows for exploration by bringing up diverse items from time to time. Of course, there might also be values the user wants to exclude from the item set. In that case, she can now dislike specific values and the score gets evenly distributed over the others. For example if a user dislikes red in the previous example , the query looks like this:

$$q = \{color_{red,blue,green,white,black}(0, 0.34, 0.34, 0.17, 0.17)\}$$

9.2.6 Integrating Interaction Design Guidelines

One of the requirements of the final prototype was that it looks similar to the baseline application, so that it is not obvious for the user which system is more sophisticated and uses more information for the calculation of the recommendations (see *subsection 9.1.1*). In general the interaction design is based on the interfaces presented in *chapter 4*. The user interfaces for the determination of the stereotype was adapted from *chapter 5*. However, we adjusted the user interfaces based on the interaction design guidelines developed in *chapter 6*. When starting the application, the stereotype is determined. Then, the user is asked to input the context conditions manually. The context determination can be seen in *figure 9.8*. The conditions for time of the day and day of the week, are not captured, as it is expected that the users are aware of these conditions subconsciously. All conditions can be selected via comboboxes as our guidelines indicate that this method is suitable for data elicitation.

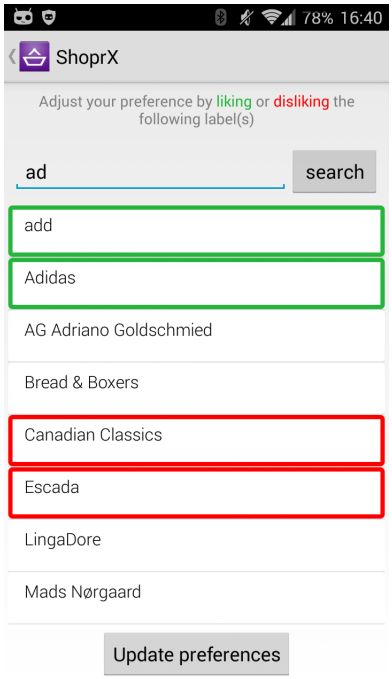
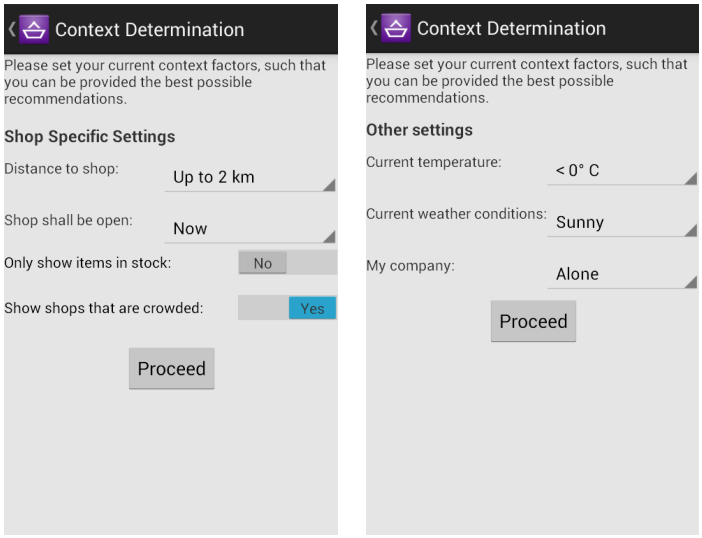


Figure 9.7: Brand setting

The recommendations are presented to the user in a grid view after she entered the context data. They represent a diverse set of items fitting the stereotype and context conditions as presented in *figure 9.1*. Results of our conducted survey in *chapter 6* show that the majority of users prefer an infinite list instead of a grid for showing recommendations. The system should present many items so that the user can scroll the



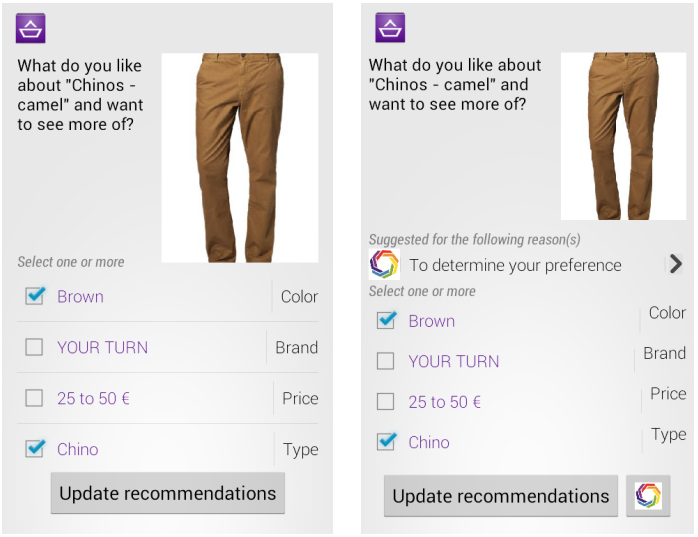
(a) Location-based context preferences

(b) Eliciting temperature, weather and company

Figure 9.8: Explicit context determination via questionnaire

list, as users appreciate the scrolling feature. On the other hand the grid layout tested in the user study performed better regarding time to select an item and average completion time for a recommendation session. Furthermore, a grid is perceived as easier to use than a list. Therefore, the grid was chosen to present items to the user. The number of displayed recommendations was set to nine, with which items can be represented in a 3×3 grid to the user. Iyengar and Lepper show that having more choices is not necessarily better than only having a few. In their study customers tried to find the preferred flavor of jam of several options at a tasting booth. Although with a small number of choices, less people were attracted, significantly more of them (and even overall) bought a product [Iyengar and Lepper, 2000]. Ricci adds that heavy scrolling can be annoying to users and that the more a user has to scroll, the less likely it is that an item is selected [Ricci, 2010]. In the opinion presented in [Reutskaja and Hogarth, 2009] the benefit of selecting the best choice minus the selection cost provide the most utility to the user. The authors therefore argue that an optimal number of items is ten. Although it was not evaluated, whether their results are also applicable to online scenarios, we assume that they hold true and present nine items to the user. If we would display ten items, the grid layout would more look like a list as it has to be presented in a 5×2 grid. This would on the one hand imply significantly more scrolling effort, but on the other hand larger pictures.

With the *thumbs up* or *thumbs down* button the user is able to criticize the items. The following screen (*figure 9.9*) gives an overview of the item in which the user is supposed to criticize by selecting specific attributes. These attributes are price, brand, clothing type and color. According to the previously developed guidelines, the user shall be allowed to individually select how she likes to criticize the items. We came up with four different

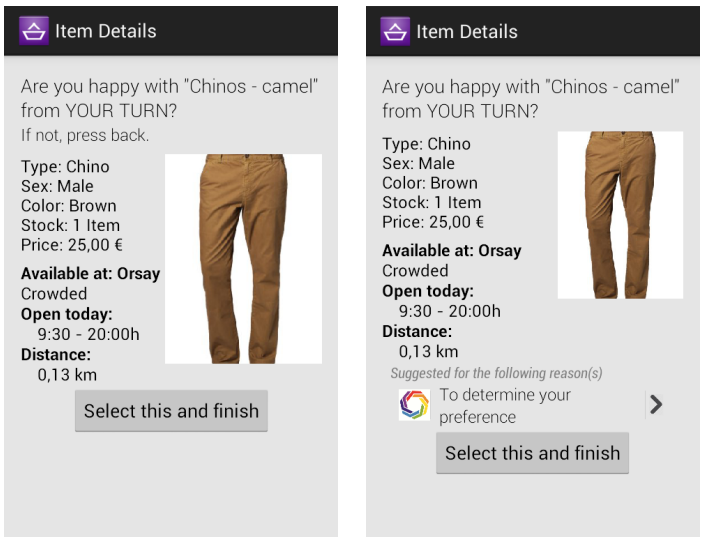


(a) Critiquing view in the baseline application

(b) Critiquing view in the final prototype

Figure 9.9: Comparison of critiquing views

methods of preference elicitation: *Rating, Like/Dislike, Positive/Negative, Replacement.* We moreover found out that in general replacement and rating perform best regarding ease of use and user satisfaction. However, replacement does not seem to be adaptable to the current scenario, as there are more than 400 different brands in the dataset. The replacement of a brand would be time-consuming and annoying, and is not likely to help in the desired exploration of items. When rating features with stars, it is difficult to transform them into critiques of features. For example what does a three star-rating (out of five) add to the preference elicitation progress? Does it say that there should be more of this feature? Even more important is, whether the used interpretation of a rating is generalizable or whether it is user-specific as it often is for ratings acquired with this method. And if the interpretation was user-specific, how could a system know this (without prior knowledge) to appropriately react on critiques? However, it is reasonable to use the like/dislike functionality, because it is easy to understand and frequently used in well-known social networks like Facebook or Google+. In the prototype a thumbs up means the user likes to see more of this feature, whereas a thumbs down can be translated that the user wants to see less of this feature. The resulting critiquing view, as well as the critiquing view of the baseline can be seen in *figure 9.9*. By clicking on an item's picture in the grid view, the user gets to another screen. On this screen, she might finally select an item. The screen now shows more information about the item and the store in which the item is available. This information should also enhance the *trust* that the user has in the recommendations as she can check whether the initial preferences (about distances to shop, crowdedness, etc.) were incorporated. The differences to the baseline version can be seen in *figure 9.10*. Again, explanations show why the item is recommended.



(a) Item view in the baseline application

(b) Item view in the final prototype

Figure 9.10: Comparison of views for item selection

Furthermore we adapted the map overview, which shows where exactly the shops are located. The information about the shop is enhanced and shows today's opening hours, the crowdedness, the shop's name, the distance to the current position and how many items (out of the current recommendations) are available at this shop. In *figure 9.11* the two versions of the map view are presented.

In order to increase the user's control over the system, we now also wanted to provide the possibility to manually adjust the degree of diversity used to calculate the recommendations. If the user clicks on the diversity icon (which is located at the bottom right of every screen) she can adjust the diversity. The system explains the effect of different diversity settings to the user in a pop up window (see *figure 9.12*). We determined three alternative values for the degree of diversity (α). The diversity α value has already been introduced in *chapter 4 (section 4.1.1)*. The standard normal diversity is still $\alpha = 0.997$. In addition, low diversity with $\alpha = 1$ and high diversity with $\alpha = 0.95$ will be added. Low diversity actually means that no diversity is considered and a similarity-based approach is applied. The high diversity α value is the same used when refocusing after a feature was disliked (more details about the Active Learning algorithm can be found in *chapter 4*).

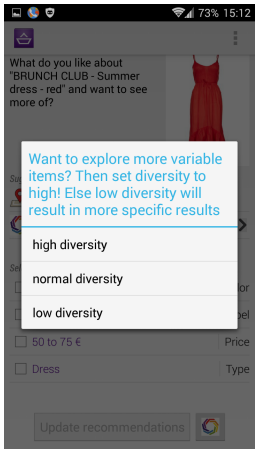
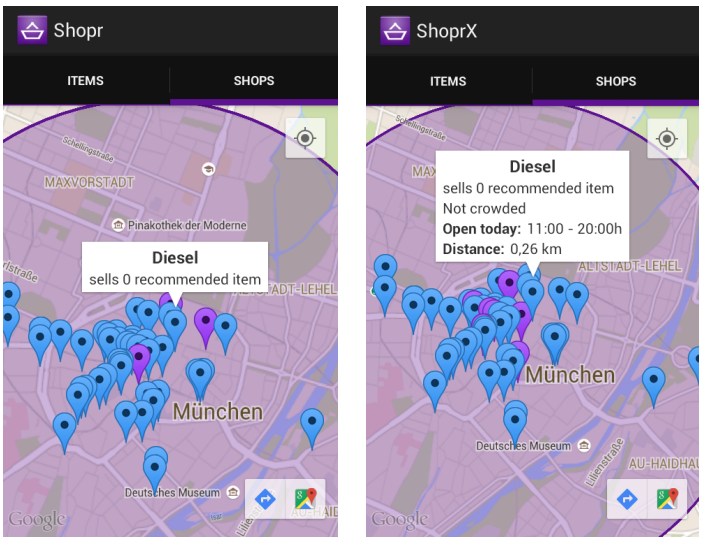


Figure 9.12: Diversity setting



(a) Shop overview in the previous version

(b) Shop overview in the baseline and final prototype

Figure 9.11: Comparison of shop overviews on map

9.3 User Study

Although the five building blocks of our conceptual framework have already been investigated and evaluated separately, we now want to find out if the interaction of all proposed concepts still creates a positive user experience when combined in one application. The user study was designed in order to test the differences in user perceptions between the final prototype and a baseline application.

9.3.1 User Study Goals

The specific goals of the user study are described in the following paragraphs. We took into account traditional measurements of a system's user experience. Some of these metrics have already been discussed in *section 2.3*.

Prediction Accuracy: Our first goal is to investigate the prediction accuracy of our mobile recommender system. It describes how well the user's preferences and clothing style are expressed by the products and reveals if our developed recommendation algorithm, which is context-aware and uses Active Learning, is accurate.

Adaptation to Context: We also want to find out if users perceive the application as context-aware. As our recommender system takes the mobile context into account and recommends products that might be interesting for the user in this particular

context, we investigate whether users perceive the recommendations as being suitable for the current context.

Intention to Return: It is also interesting to find out if the users like the application so much that they would use it again when going shopping in the future. In other words, we want to measure the intention to return to the system.

Decision Effort: Another main goal is to investigate the effectiveness of our application, compared to the baseline. For this purpose, the period between the start of the recommendation process (after inputting the stereotype and context data) and the selection of a product as well as the number of critiquing cycles that were needed, are measured.

Item Frequency: As one of the goals of our recommender system is to inspire the user in an exploratory scenario such as going shopping, we want to increase the number of different items that are shown. We therefore count how often each item is suggested to the user until the final item is selected.

Transparency: Due to the interactive explanations we want to measure if the sophisticated prototype is more transparent than the baseline. A transparent system lets the user reproduce how the recommendation process works.

Scrutability: The application explains recommendations and allows users to interact with the explanations. The reason for this feature is that we want to give the user control over the system. We therefore want to evaluate if the user feels capable of correcting wrong assumptions made by the system.

Trust: Recommender systems use personal data to personalize the recommendations. People might be concerned of the usage of sensitive information. One goal of explanations for recommendations is to reveal why and how this information is used in order to build trust. We therefore want to find out if our system is trustworthy.

Persuasiveness: Another goal of implementing explanations into a recommender system is to change a person's attitude in a predetermined way. In this user study we will measure if the participants are more likely to accept the recommendations compared to the baseline.

System Preference: The main goal of developing our prototype was to achieve a higher user experience than with the baseline system. Although the previously described evaluation metrics already give an indication about the user experience, we conclude the survey by asking the users which application they prefer.

In addition to these evaluation goals we also encourage users to give informal feedback on the applications' designs and on technical features in general. How we aim to measure these goals within our user study is described in the following.

Figure 9.13: Tool to generate a user's scenario

9.3.2 Setup

The tests were performed with two participants at a time, using a *Samsung galaxy S2* and *S3* with 4.3 and 4.8 inch screen size. Both were running on *CyanogenMod* for *Android* 4.4 with version "nightly". The user study is designed as a supervised within-subjects user survey to minimize the number of survey participants and improve the comparability between the applications. Each user tests both applications (the baseline system and the final prototype) and answers a questionnaire afterwards. Both applications look very similar to each other, however, only the final prototype uses context-awareness as well as interactive explanations. Which system is tested first is flipped in between subjects so that a bias because of learning effects could be reduced. The participants are asked to imagine being in the scenario, the tool generated for them, whereby the location is always Munich. The participant's task is to find one item only, which they would like to try on. As soon as the users have found a suitable item, they are asked to select it, such that they can finish the test and answer the corresponding questionnaire. The target population of this application are young smartphone users that like to go shopping. In the user survey qualitative and quantitative data are collected. Qualitative data is measured via a questionnaire. It mainly consists of statements, the user should assess on a 7-point Likert

	Baseline		Final prototype		p value	V value
	mean	stdev	mean	stdev		
Perceived accuracy	2.71	1.39	2.34	1.14	**.009**	1807
Perceived context-awareness	2.82	1.61	2.66	1.44	0.54	1346
Intention to return	3.06	1.54	1.26	1.26	**.002**	1563
Time	179 s	140.16	182 s	115.86	0.45	2302.5
Cycles	7.34	6.6	6.1	4.92	0.11	2393.5
Item frequency	1.46	4.04	1.24	3.03	**<.001**	285253.5
Transparency	2.17	1.56	2.03	1.26	0.40	590.5
Scrutability	2.96	1.58	2.41	1.39	**.006**	1985.5
Trust	2.99	1.21	2.93	1.31	0.49	355.5
Persuasiveness	2.53	1.47	2.72	1.35	0.11	1088.5

Table 9.1: The means of some important measured values comparing both variations of the system.

scale (this time we used a reversed scale from 1, strongly agree to 7, strongly disagree), e.g., how satisfied the user is with the recommendations and the application in general. The complete questionnaire is shown in *Appendix F.4*. The quantitative data is directly measured within the application and includes the number of critiquing cycles, the time between viewing the first set of recommendations and selecting an item, and the item diversity. Before the user starts using the application, a scenario describing the user's location, weather and company is generated for her (see *figure 9.13*). The participants are asked to actively select their context in the application and imagine it. This scenario is visually displayed to the users throughout the whole survey on a computer screen directly in front of them. The context conditions not mentioned in the scenario description, such as the crowdedness, can be selected by the user based on her own preferences.

9.3.3 Results

All in all 100 participants (48 females, 52 males) took part in the user survey in the 17-to-30 age range. The answers to the 7-point Likert scale statements (from 1, strongly agree to 7, strongly disagree) in this work either followed a positively or negatively skewed distribution and are ordinal scaled instead of interval scaled. Therefore, a two-tailed paired Wilcoxon signed rank test is executed, rather than a paired t-test, to detect whether there are any significant differences between the distributions. The results of the two-sided tests are reported by stating a V and a p value. The V is the sum of ranks assigned to differences with a positive sign. Therefore, a higher V stands for higher differences in the user's decisions. The p value defines how significant the results are at a significance level of 0.05. In general, we evaluate whether the null hypothesis is likely to be true. The means, the

standard deviation (stdev) as well as the V and p values of the most important metrics of the two systems are shown in *table 9.1* (significant results are printed bold).

Prediction Accuracy

In order to test the user's perceived prediction accuracy, we asked if the recommended products fitted the individual preferences. The baseline application's mean is 2.71 whereas the mean of the final prototype is 2.34 ($Median = 2$ for both systems). The Wilcoxon signed rank test reveals, that the recommendations of the more sophisticated approach fitted significantly better to the user's preferences than the baseline's recommendations ($V = 1807$, $p = 0.009$). These results are illustrated in *figure 9.14*.

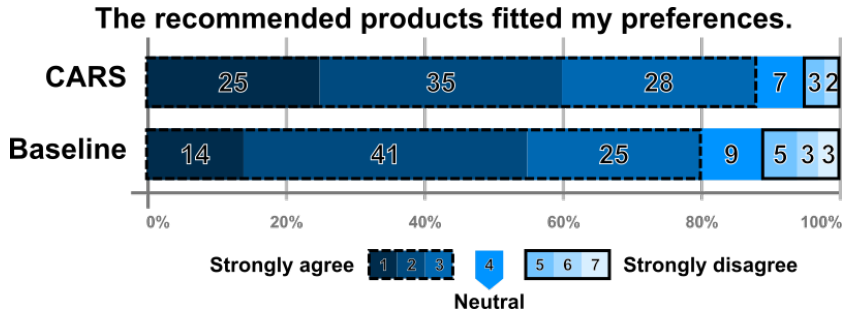


Figure 9.14: Distribution of ratings for prediction accuracy on a 7-point Likert scale

Adaptation to Context

The context-awareness of the applications is evaluated by asking whether the products were in line with the provided scenario. The baseline application's mean is 2.82 whereas the mean of the new prototype is 2.66 ($Median = 2$ for both applications). The Wilcoxon signed rank test shows $V = 1346$, $p = 0.54$. This means that the users did not perceive any of the systems as being more context-aware than the other (see *figure 9.15* for a graphical representation).

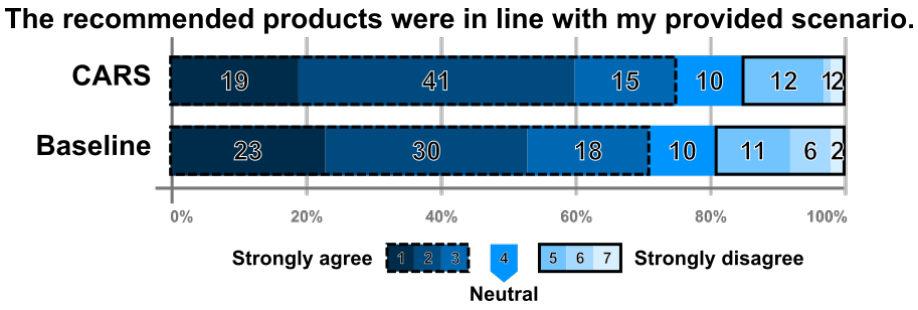


Figure 9.15: Distribution of ratings for adaptation to context on a 7-point Likert scale

Intention to Return

When asking the users whether they are likely to use the application again, the users stated that they are significantly more likely to use the final prototype ($Median = 2$, $Mean = 2.64$) again, than the baseline ($Median = 3$, $Mean = 3.06$) application ($V = 1563$, $p = 0.002$), as illustrated in *figure 9.16*.

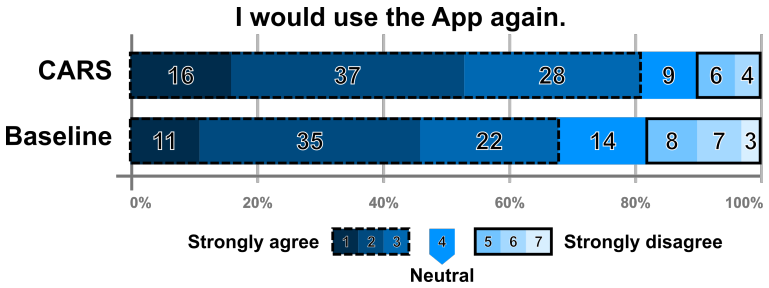


Figure 9.16: Distribution of ratings for intention to return on a 7-point Likert scale

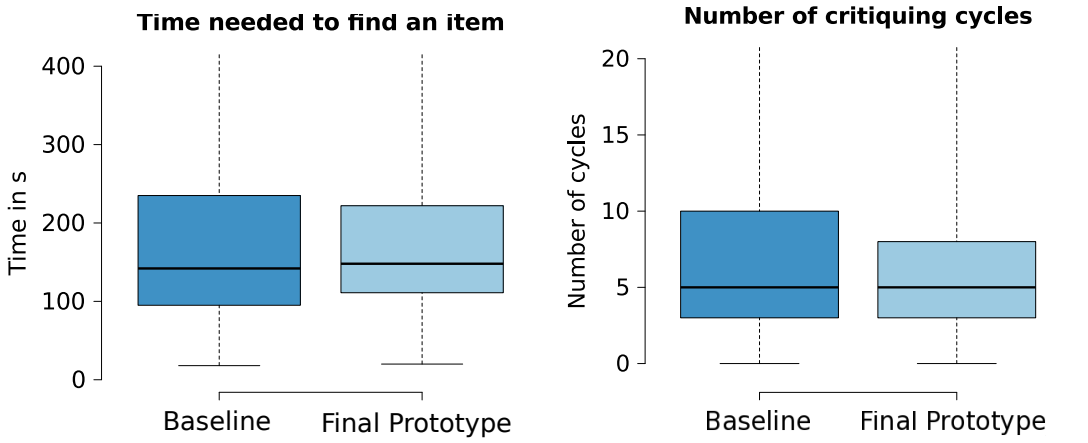
Decision Effort

The maximum time needed to find an item in the baseline application was 867 seconds ($Median = 142s$, $Mean = 179s$) and in the more sophisticated application 697 seconds ($Median = 149s$, $Mean = 182s$). The time needed to select an item is not significantly different between the applications ($V = 2302.5$, $p = 0.45$). *Figure 9.17 a* illustrates this result.

Another measure for the effectiveness of the recommendation algorithm is the number of critiquing cycles until an item was selected. Participants completed their task in average 1,24 cycles less using th final prototype ($Median = 5$, $Mean = 6.1$ with the new approach, $Median = 5$, $Mean = 7.34$ with the baseline system). Again a Wilcoxon signed rank test was executed ($V = 2393.5$, $p = 0.11$). However, the result is not significant, meaning that the null hypothesis cannot be rejected (see *figure 9.17 b*).

Item Frequency

One of the goals of the final application was to reduce the number of times an individual item is shown (*item frequency*) and thus increase the number of different items (*item coverage*). All in all the baseline application showed 7506 (1690 different; 22.5 % unique) and the more sophisticated application 6390 (1754 different; 27.4 % unique) items. We measured every time that an item was displayed to any user. The maximum number of times an item was shown was 115 ($Median = 1.46$, $Mean = 0$) for the baseline application and 53 ($Median = 1.24$, $Mean = 0$) for the final prototype. A Wilcoxon signed rank test reveals that there is a significant difference between the samples ($V = 285253.5$, $p < 0.001$),



(a) Required time to find an item (b) Number of critiquing cycles to find an item

Figure 9.17: Box plots of the time in seconds (left) required to complete a session and number of critiquing cycles (the maximum has been omitted for space reasons)

meaning that the new approach showed items significantly less frequent than the baseline. Although the new application showed less items overall, more different items have been shown. This indicates that the recommended items have been more diverse.

Transparency

The most obvious goal of implementing explanations in a recommender system is transparency. We therefore asked the participants if they understood how the recommendations were created. The answers from the questionnaire did not indicate higher transparency ($V = 590.5$, $p = 0.40$). The baseline had a *Median* of 2 and a *Mean* of 2.17, the more sophisticated prototype a *Median* of 2 and a *Mean* of 2.03. *Figure 9.18* illustrates these results. A reason why there is no significant difference might be that as Chen and Pu point out, "critique suggestions can perform as explanations and help users be familiar with the product domain and the relationship between attributes" [Chen and Pu, 2011, p. 135]. So the user knows which attributes are used as input for the recommendation algorithm.

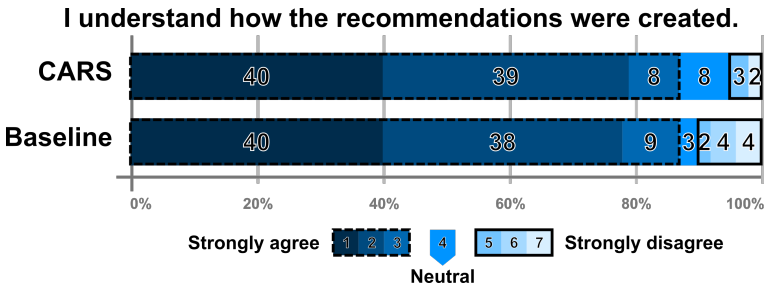


Figure 9.18: Distribution of ratings for transparency on a 7-point Likert scale

Scrutability

We also wanted to find out whether the users were more satisfied with how they could influence the recommendations. It is expected that the more sophisticated prototype performs better, due to its adapted recommendation algorithm (adding novelty and diversity) and the integrated explanations. We therefore asked the users if they could influence the exploring process according to their expectations. This statement should give an idea on how well the system reacted to critiques, but as well adapts to the specific user. With $Median = 2.5$, $Mean = 2.96$ for the baseline and $Median = 2$, $Mean = 2.41$ for the new approach, there is a significant difference between the two systems: $V = 1985.5$, $p = 0.006$, as can be seen in *figure 9.19*. As a result the null hypothesis has to be rejected, meaning that the users were better able to influence the recommendation process, when they used the new prototype.

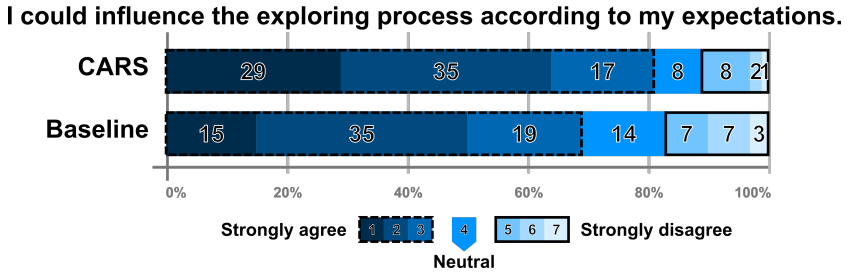


Figure 9.19: Distribution of ratings for scrutability on a 7-point Likert scale

Trust

In order to measure if users trust the system, we asked if the system handles information trustworthy. People trusted both systems, however the final prototype is not significantly more trustworthy compared to the baseline. The baseline has a $Median$ of 3 and a $Mean$ of 2.99, the new prototype has a $Median$ of 3 and a $Mean$ of 2.93. The Wilcoxon signed rank test shows $V = 335.5$, $p = 0.49$ so that the null hypotheses can not be rejected (see also *figure 9.20*).

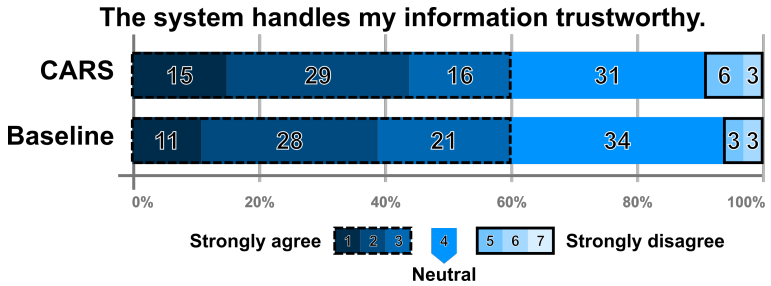


Figure 9.20: Distribution of ratings for trustworthiness on a 7-point Likert scale

This could be explained based on the usage of sensitive information. Explanations tell the user which information is used and for what. Thus, clients might think that information is handled more trustworthy than in a blackbox system. In this approach, the context information was neither real nor really sensitive, so it is not surprising that the trustworthiness was not rated significantly better.

Persuasiveness

Measuring perceived persuasiveness is problematic, because people do not like to be manipulated or influenced. In consequence, they might have problems being completely honest about it. The question to measure it has to be chosen carefully, so it does not sound negative and causes those side effects. We therefore asked the users if the system helped to make a decision. Although the participants rated the new application ($Median = 2$, $Mean = 2.53$) better than the baseline ($Median = 2$, $Mean = 2.72$), this difference is not significant ($V = 1088.5$, $p = 0.11$), as can be seen in *figure 9.21*. A possible reason for this is the quite high score for both applications.

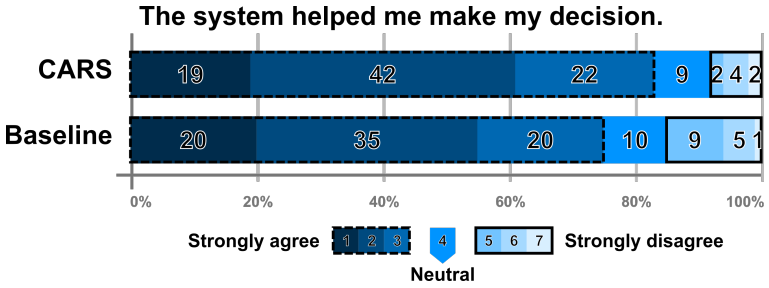


Figure 9.21: Distribution of ratings for persuasiveness on a 7-point Likert scale

System Preference

Overall, 59 participants reported that they prefer the final prototype (see *figure 9.22*). This are significantly more compared to a random distribution of answers as a chi-squared test reveals ($\mathcal{X}^2 = 30.38$, with 2 *df* [degrees of freedom], $p < 0.001$).

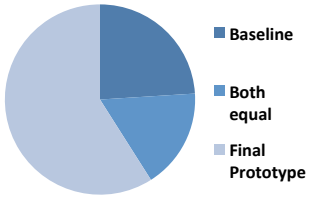


Figure 9.22: Preferred variant

9.4 Discussion of the Results

in summary, the results of the user study were very satisfying and led to many valuable insights into factors that influence the user experience of mobile recommender systems. The user study covered **100 participants** that tested a simple mobile shopping recommender system (*baseline application*) against a more sophisticated application (*final prototype*).

Both systems allow users to criticize items based on their preferences and use stereotypes to already provide personalized recommendations in the beginning to overcome the cold start problem. However only the final prototype is context-aware and generates interactive explanations. The participants tested each application and gave their impression of the systems afterwards. It was investigated, whether the final prototype is able to provide better recommendations and improves the user experience, compared to the baseline application.

59 test participants preferred the new approach over the baseline. Results show that the recommendations of the more sophisticated prototype matched their preferences and clothing style significantly better. Furthermore, they evaluated that the new application in general provides **better recommendations** and that they would rather reuse it than the baseline application. Whenever a dimension was used for explanations, the users were allowed to correct the assumptions of the final prototype. Only 34% of the participants used the feature to set their preference directly, and most of them needed a high amount of cycles to find an item. This proves most users are satisfied with the algorithm and only need those tools in rare cases, for example when they wanted to switch quickly to a certain clothing type. Still, it is very clear, that control over the system is very much appreciated. The comments on both applications prove that as well. The test participants stated that the final prototype is able to make better recommendations and that they were able to influence the exploring process better in the new system. They perceived **more control** of the new approach to correct wrong assumptions that were made.

The measured data in the system shows no significant differences between the baseline and the final prototype regarding the **time** and the **number of critiquing cycles** needed until an item was selected. However, the new system has a **higher item frequency**. It showed the same items significantly less often than the baseline application. This means, that it is able to show a more diverse item set than the baseline and can have an inspiring effect, which is in particular important in an exploratory scenario such as mobile shopping. The comments of the users show, that although the diversity of the recommender algorithm has been adapted, diversity still is a major issue for recommender systems, as 11 participants (five for the final prototype and six for the baseline) stated that the recommendations were too specific to the last search queries. Hence, even though McGinty and Smyth provided a good algorithm that supports critiquing for the diversity of recommendations, this algorithm could still be improved [McGinty and Smyth, 2003a]. The constance in time could be explained by the necessity to read more explanations, and interact with those. The number of critiquing cycles could have stayed constant, because the users liked the system more and therefore wanted to make absolutely sure that they find a really good item. Hence, it should be researched whether there are other objective measures for the prediction accuracy. One could argue that the time a user watches an item also could measure the satisfaction with items. Much scrolling could indicate that the user either cannot decide between two items or is unsatisfied with the recommendations and tries to find out which item she likes. It could be distinguished between those two possibilities by evaluating whether the next critique is positive or negative. It can also be assumed, that after having used the application several times, the user needs less and less

time to read the explanations and that even a higher time consumption does not negatively affect the user experience.

As for the **context-awareness** the users did not perceive any of the systems as being more context-aware than the other. As the algorithm for recommending items was only altered in the way that it additionally includes a post-filtering, which filters items for context reasons and does not take the user's preferences into account, it is not surprising that the users did not perceive the system as creating better recommendations according to their clothing style. Although the users did not perceive any significant improvement in the context-awareness of the recommendations, they stated that the recommendations provided by the final prototype are significantly better than those of the baseline application. One explanation might be that the users could not relate the improved recommendations to the context, because the context-awareness is only perceived subconsciously. It should be investigated in the future, if users are able to perceive whether a recommendation is context-aware or not, without giving hints via the user interface and further evaluate whether they perceive context-awareness consciously or subconsciously. Overall nine participants explicitly stated that either the final prototype was more context-aware or the baseline was not context-aware. Some of these participants also stated that they appreciated that the application recommends, e.g., shorts, shirts and swimwear when it is very hot. These results are in line with [Adomavicius and Tuzhilin, 2005], who found out that context matters in e-commerce, if it is able to clearly divide the rating of an item depending on the context. Obviously temperature is a context factor, that is able to divide an item's rating as some participants stated that they do not want to shop for sweatshirts at 30° C. In addition to these results it could be shown, that a baseline content-based recommender system can profit from a context-aware approach to provide better recommendations in general and not just when the item space can be clearly divided. The given results also support the hypothesis of [Panniello et al., 2014] that post-filtering can lead to significant improvements in the recommendation quality. It enhances this hypothesis in the way that post-filtering does not only work well for one-time recommendations, which are chosen from a dataset, but also in an iterative approach. Similar to the approaches of acquiring context-relevance for points of interest presented in [Baltrunas et al., 2011], we asked the users to imagine being in a certain context. As the users were asked to imagine several context factors at once, it seems that the influence or the awareness of these context factors decreases. Therefore, it should be investigated to which degree a mere imagination of context conditions is able to substitute being really in the specific context. Results from psychology research of Gregory et al. show that imagining certain events, makes persons believe more strongly that these events might happen to them. The authors also found out that people imagining certain actions are more likely to execute them in practice afterwards [Gregory et al., 1982]. Therefore, it might be possible that imagining context is in fact applicable to test a context-aware application. However, we did not find any studies proving this.

Of course there are **limitations** of the results of this laboratory experiment. The first is that the results drawn via this experiment are mainly valid for students. They might also be valid for other people in the 15-to-35 age range. A further generalization of these

results might be inadequate. A second limitation is that the users could not be put into the real context scenario and had to imagine it instead. A more realistic setting, where users would have tested the applications while going shopping in a city and where real contextual information from built-in sensors would have been used, might have performed even more significant results. However, despite these limitations and shortcomings, the final, more sophisticated prototype proved to provide a positive user experience.

All in all it was shown that the final prototype provides significantly better recommendations than the baseline application. Even so, this difference was only perceived by the users and not directly recognizable from objective measurements such as time and critiquing cycles, the user experience of the new application was perceived as more positive. The improvement in the perceived accuracy of recommendations could not be linked to the scenario, but instead was linked as matching to the user's preferences and her clothing style, for which the calculation was unchanged in both versions of the application. In general, users liked the interaction design as well as the ability to criticize the suggested items. Both applications were designed based on the guidelines established in *chapter 6*. A main conclusion is that a mobile recommender system based on the considerations of our framework helps people to filter the necessary information to complete a specific task and even improves the user experience. However, the user study was conducted in a laboratory setting and some of the used data was artificial. For a final assessment of the proposed framework, a long term study in a real shopping scenario needs to be undertaken. Moreover, we can imagine more scenarios where such a mobile recommender system could support the user, such as a mobile restaurant or points of interest recommender system. Future research should evaluate the proposed framework also in other mobile scenarios.

9.5 Summary

Within this chapter we presented the results of our final user study. We first explained the development of our final prototype and pointed out how it differs from the baseline and the previously evaluated approaches. Then, the goals of our user study were described as well as the user study setup. 100 participants evaluated the two prototypes in a laboratory setting. The results of the user study were very promising. We showed that the recommendations of the final prototype are perceived as better compared to the baseline. However, the context-awareness of the more sophisticated application was only perceived subconsciously. Although there was no significant difference between the time and the number of cycles that were needed until the task was finished, the user experience of the sophisticated prototype was improved. In particular users appreciated the interactive explanations and the associated control over the system. Both system designs were ought to look similar so that it was not obvious which prototype is more sophisticated and takes additional information into account. Participants liked the overall interaction design as well as the possibility to criticize the items in both applications a lot. The chapter concluded with a critical discussion of the results.

10

Conclusion

In this thesis, several aspects of a user-friendly mobile recommender system have been revealed. In the following section (1), we will summarize the research endeavors and results of each chapter. We then discuss technological constraints of our prototype, challenges that need to be considered when applying our proposed framework and interesting observations we made during the final evaluation (2). In this context, we also emphasize the main contributions of this work. In the last section (3) we present several ideas for future work. We come up with different application scenarios to which our conceptual framework could be applied and discuss other suitable recommendation approaches and novel technologies that could be used in order to support a positive user experience.

10.1 Summary

In the introduction (**chapter 1**), we explained the motivation of the thesis. We furthermore defined the term "user experience" and introduced the main research question. We then described our methodology that is based on the design-science research guidelines by [von Alan et al., 2004]. Here, we also pointed out the undertaken activities in order to follow this proposed methodology. Next, we summarized the main contributions of our work and gave an overview of each chapter.

In **chapter 2** we introduced the fundamental terms and concepts regarding (mobile) recommender systems. We gave a general definition of the term "recommender system" and discussed its history. We then explained the functionality of a recommender system and listed different approaches. In particular we described the concept behind a conversational recommender system since our proposed solution is based on that approach. We furthermore discussed evaluation techniques which we used in order to test the user experience of our developed prototypes. In the final section of this chapter we classified mobile recommender systems and discussed existing approaches and relevant results concerning our research question.

Based on this foundation, we investigated in **chapter 3** which human-computer interaction (HCI) aspects should be considered when developing a mobile recommender system with a positive user experience. Since the user-friendliness of a system plays an important role for the success of a system, we analyzed existing recommender systems providing a positive user experience. Those aspects that turned out to improve the user experience of a mobile recommender system were then integrated into a conceptual framework. The building blocks of this framework represent the research gaps in the scope of mobile recommender systems with a positive user experience that were separately investigated in the following chapters. The chapter concluded by presenting the selected application scenario of our prototypes and alternatives.

Chapter 4 described the implementation and evaluation of a mobile shopping recommender system using Active Learning. We explained why we selected such an approach and the details of our prototype - a mobile shopping recommender system that uses Active Learning by eliciting critiques from the user. Within a following user study, we compared two different variants of an Active Learning algorithm: One based on diversity, one based on similarity. Results showed that the diversity-based approach provided a higher user experience and was preferred by the majority of users.

Chapter 5 introduced the cold-start problem in a mobile recommender system. We presented a solution to this problem that is based on stereotypes and allows the creation of a user model already in the beginning of the process. We were therefore able to deliver accurate personalized recommendations right away. In our prototype we first implemented navigation by asking to determine the user's stereotype and to deliver personalized recommendations. Navigation by proposing was then used to elicit the user's critiques and to refine the user model. Our user study showed that the prototype based on stereotypes outperformed the system without a stereotype-based logic.

In **Chapter 6** we investigated the interaction design of a mobile shopping recommender system. Different challenges have to be overcome when designing a mobile recommender system compared to a desktop system, e.g., regarding the smaller screen size or input methods. Moreover mobile recommendations can be improved by taking information about the mobile context into account. We therefore investigated a complete interaction design process. First, specific requirements of the interaction design of a mobile recommender system were analyzed. Then we designed solutions that meet these requirements and produced a low- and higher-fidelity prototype based on this. After having conducted a two-part evaluation, we were able to come up with interaction guidelines that help improving the user experience of a mobile recommender system.

Chapter 7 focused on the integration of context-aware information into a mobile recommender system. We wanted to find out if the integration of mobile context improves mobile recommendations. Therefore we first conducted a survey to assess the influence of specific context factors on the shopping behavior. We found out that especially the context information weather, budget and shopping intent have the potential to improve the accuracy, efficiency as well as the user experience when considered by a mobile recommender system. We then implemented a mobile shopping recommender system that takes

these context factors into account. A following evaluation showed that our context-aware prototype performed better than the application that was not context-aware.

In **Chapter 8** we came up with a concept to provide interactive explanations in a mobile recommender system. Explanations in a recommender system have the potential to help users to make better decisions and also provide a higher transparency, compared to a recommender system without explanations. We considered the characteristics of a mobile device and came up with a solution that allows the generation of interactive explanations. Whenever the system makes wrong assumptions about the preferences, the user may correct them by interacting with the explanations. The proposed concept was implemented into a mobile shopping recommender system. The users appreciated this interactive approach a lot and results of the user study showed that the user experience was improved, compared to a baseline system.

Chapter 9 described the goals, the set up and the implementation of the final user study. We wanted to test whether the different features developed during the investigation of the building blocks, still provided a positive user experience when combined in one system. We therefore developed a prototype that implements all previously defined concepts and tested it within a user study of hundred participants. Results of the study were presented and critically discussed. We were able to show that our final prototype provided a more positive user experience compared to the baseline system so that the usefulness of our conceptual framework could be proved.

10.2 Discussion

Our research approach first involved a systematic analysis of related work in the area of HCI and (mobile) recommender systems and based on that the development of a conceptual framework that promises to generate a positive user experience when applied in a mobile recommender system (see *chapter 3*). The evaluations of the individual building blocks of the framework have shown that each aspect individually contributed to the improvement of the user experience of our prototypes. Based on the results of our pre-studies, we made some adaptations to the final prototype and also the combination of the building blocks was a success. Within our different user studies, we wanted to find out if the conceptual framework needs to be adapted to a greater extent (e.g., by excluding one or several aspects), but none of the building blocks could be considered as unnecessary. However, it might be possible that the aspects we excluded already in the beginning, such as the application of different user modeling approaches or the consideration of implicit feedback (see *subsection 3.3*) might have achieved positive results as well.

Although the development of the final prototype consisted of an iterative process that took the results of five separate pre-studies into account, some technical limitations were still present in the final evaluation experiment. We summarize them in the following (*see also section 9.4* for a detailed discussion of the final user study). The final user study was conducted in a laboratory setting. A more realistic setting, where users would have

interacted with the application while going shopping in a city, might have performed even more significant results. Moreover, since we were not able to establish cooperations with clothing retailers, the database was artificial and the fashion items were randomly assigned to different shops in Munich. Also the context-scenario was predefined and users were asked to imagine being in that scenario. Although we tried to create this scenario as realistic as possible (e.g., by consistently showing suitable pictures) it has to be found out if this experimental setting affected the results in any way. Moreover, the participants of our user study were mainly students in the 18-to-27 age range, as we considered this as the target group of such a recommender system. Therefore the results drawn via the experiments are mainly valid for this age group and profession. However, a generalization of these results for different target groups might be inadequate. Another limitation of our user study is that the implemented stereotype user model was static. For more accurate results, machine learning techniques could be used that automatically assign users to dynamic stereotypes just based on their interactions with the application.

Developers of mobile recommender systems need to consider some challenges when applying our proposed conceptual framework. First, they need to generate a dataset that is big enough for an efficient Active Learning recommendation strategy. This is important to allow unlimited interaction cycles without showing items that do not match the user's explicitly stated preferences. Also the diversity parameter (α) depends on the dataset and needs to be configured based on pre-tests. Moreover, the interaction guidelines were evaluated for a smartphone. Whenever a different mobile device (e.g., tablet) should be used, slightly different interaction guidelines might apply. Regarding the application scenario, an exploratory scenario might benefit of our proposed framework the most. The reason is that our recommendation algorithm is designed to inspire users and was not adapted to a scenario where the user already has a specific product in mind. The necessary detailedness of the explanations also depends on the application scenario and its complexity. We suggest a stereotype user model to overcome the cold-start problem at the beginning of a recommendation process. However, when applying our proposed user model for another application scenario, our developed fashion stereotypes need to be replaced by scenario-specific ones. Our proposed approach to determine which mobile context should be taken into account by a shopping recommender algorithm, can also be adapted for other scenarios.

Despite these limitations and challenges, the overall results are very promising. The five pre-studies already delivered good results. We first developed a recommendation algorithm that takes Active Learning into account and allows users to criticize items according to their preferences. The diversity-based algorithm delivered more accurate recommendations than the similarity-based one and created also an improved user experience. We then investigated a stereotype-based user model to overcome the cold-start problem. A combination of navigation by asking and navigation by proposing allowed the determination of the user's fashion stereotype and delivered more personalized results than a system without a stereotype-based user model. We conducted two different user studies to deliver interaction design guidelines for a mobile shopping recommender system. Considering all these guidelines supports the development of user-friendly and intuitive user interfaces so that a positive user experience can be provided. In another study we investigated mo-

bile context. We first analyzed those context factors that have the potential to improve mobile recommendations and then implemented a context-aware prototype. Results of our user study showed that also context-awareness improves the user experience of a mobile recommender system. The last established concept covered interactive explanations. We used a smartphone's specific interactivity aspects in order to allow users to interact with the explanations. Also the implementation of this concept showed the improved user experience in a following evaluation. Finally we developed a prototype that implements the five different concepts for a shopping scenario. Within a user study of 100 people we were able to show that the combination of all these building blocks in one system and the interaction with each other provides a positive user experience when using the mobile recommender system. The users appreciated the application's design as well as the control over the system when interacting with the explanations. Using stereotypes provided personalized recommendations already in the beginning of the process. The participant's explicit critiques as well as the consideration of the current context delivered even more accurate recommendations so that all of the users quickly found a product that matched their preferences.

In conclusion, the main **contributions** of this thesis are the following:

Contribution 1: A comprehensive analysis of the **theoretical background** and related work of mobile recommender systems was made. The main focus hereby was on the user experience. We gained knowledge about different aspects of a mobile recommender system that positively influence the user experience. These aspects that have not yet been investigated thoroughly, represent the five building blocks of our proposed **conceptual framework**: Active Learning, user modeling, presentation, context and interactive explanations.

Contribution 2: For each of these building blocks, deeper research was performed and five different **concepts** for the **improvement** of a mobile recommender system's **user experience** were established. We proposed an efficient algorithm that takes Active Learning into account, concrete presentation guidelines, a suitable stereotype user model to overcome the cold-start problem, an optimized recommendation algorithm that considers mobile context and a concept to automatically generate interactive explanations.

Contribution 3: We implemented five different **prototypes** for each of these concepts. Within separate **user studies** they were compared to the baseline system. The findings of each evaluation iteratively influenced the following implementations and were also considered for the development of the final prototype. The implementation of the individual concepts already improved the user experience of each system, however we also wanted to evaluate if the interaction of all these building blocks still creates a positive user experience when combined in one system.

Contribution 4: The **final prototype** represents a mobile shopping recommender system that implements all evaluated concepts. We conducted a **final user study** with 100 participants. The results were very promising and proved that the user experience of the final prototype was improved and outperformed the baseline. The interaction design was **intuitive** and the recommendations were **accurate**. Moreover, participants highly appreciated the **control over the system** due to the interactive explanations. The integration of fashion stereotypes allowed personalized **recommendations already in the beginning**. The context-awareness was only perceived subconsciously.

10.3 Prospects for Future Work

Based on these results, there are several prospects for future work that can be conducted by applying the presented conceptual framework.

Regarding the commercialization of the application, a real dataset instead of the artificial one needs to be obtained from retail stores, including the real time stock information for each clothing item, the crowdedness of the store, and so on. Cooperations with closing retailers would be necessary which is difficult because retailers are usually quite sensitive to the provision of such data. This then becomes a business model problem and added value for retailers has to be created, e.g., by proving, that this kind of application attracts more customers to the store. Also self-service technologies that allow customers to produce a service without the need of an employee create a high shopping experience and might gain increasing research interest [Zagel, 2015]. For a more successful cooperation with retailers we can also imagine a more proactive approach. For example if a user passes a store that offers items she could like, time and location-dependent push-notifications could notify the user. It is also possible to provide an on demand recommendation when the user is standing in front of a store (e.g., "This store sells these particular items that might be interesting for you"). Newest applications even offer customers rewards for entering stores or scanning products. For example the application *Shopkick* uses Apples iBeacon technology [Lunden, 2013] to detect the consumer's presence in stores. A more sophisticated mobile shopping recommender system could also not only offer individual items but whole outfits, so that the user might be willing to purchase more. One approach towards this direction was presented by [Chen, 2013] who applied *crowdsourcing* to support users in decision making by letting experts pick matching clothes.

With new technologies entering the market rapidly, the overall user experience of a mobile recommender system can be further improved. This does not only affect commercial products such as more sophisticated smartphones, but also smartwatches or tablets which could be imagined to be used as recommender systems [Hammer et al.,]. Also the recently launched *Google Glass* might be a platform to use augmented reality for a novel recommendation approach with a positive user experience [Zhang et al., 2013]. Adapting the proposed framework to such mobile devices requires further research, in particular

regarding the interaction design. Additional features of today's smartphones can be utilized to simplify the interaction with a recommender system and hence improve the user experience. We already investigated the option to take a picture of an item as a starting point for an exploratory search process [Woerndl and Lamche, 2015]. This is in particular useful if a user is unable to describe what she is looking for by keyword-based search. However, more advanced image recognition techniques are needed for a more sophisticated approach. Those techniques might also help to describe items in a dataset more precisely and recognize e.g., color gradients, labels and the item's specific shape.

Moreover technologies such as cloud services could be used to address the issues related to commercial readiness of mobile recommender systems. By migrating the recommendation algorithm and main dataset store to a cloud based solution, the mobile application would then represent a mere client that catches relevant data on the device to reduce network traffic costs. In that way, algorithm complexity could be increased, because the system is not anymore constricted by the bounds of mobile hardware. However, privacy concerns may need to be addressed especially regarding the transmission of context data of the user and the establishment of a permanent profile at an off-site location, which is not controlled by the user. The authors Ricci and Polatidis and Georgiadis emphasize that security and privacy issues have to be taken into account by developers of mobile recommender systems [Ricci, 2010], [Polatidis and Georgiadis, 2013]. User data should be protected by encrypting user data and network connections. More research is needed to successfully prevent other services to be able to infer information about individuals, e.g., by obfuscating the sent information. Besides taking care of the users' privacy, Polatidis and Georgiadis also suggest to take the users' social network profiles into account [Polatidis and Georgiadis, 2013]. However, collecting the necessary data for more accurate recommendations is a challenging task and again privacy issues have to be considered.

Finally, also other application scenarios could benefit of this conceptual framework. In addition to well-known application scenarios such as mobile recommender systems for restaurants or points of interest, the persuasive aspect of recommender systems has gained increasing attention. Persuasive recommender systems are more likely to change a person's attitude or behavior [Fogg, 1998]. We suggest that this characteristic of a mobile recommender system should be used to convince users to a healthier or more sustainable lifestyle. Since our previous prototypes were tailored to a shopping scenario, we now want to find out if our results can be generalized. We are currently investigating two persuasive systems that apply our conceptual framework: A mobile recommender system for recipes as well as for private health insurances. Moreover, a mobile recommender system that selects sustainable products (e.g., fair trade groceries) or presents convenient energy-saving measures can be imagined. Such systems make the user's life not only easier but also supports a healthier or more sustainable lifestyle. Right now, only few research endeavors exist in this area (e.g., [Emrich et al., 2014]). Hence, any contribution in this direction is considered as valuable and hopefully will gain increasing research interest.

Appendix A

Active Learning User Study Survey

This is not the actual survey form, the questions are, however, identical in wording and their possibilities for responses.

A.1 Demographic Questions

The survey starts with some demographic questions to allow statements about the sampled population.

Question How old are you?

Answer Text field

Question What is your primary profession?

Answer Student | Employee | Self-Employed | Other: [*Text field*]

Question Do you have online shopping experience?

Explanation *Browsing and ordering from Amazon, eBay, hardwareversand or other online shops.*

Answer Yes | No

Question Do you shop from your mobile device?

Explanation *Through an app or the web browser on your phone, tablet or a similar device.*

Answer Yes | No

Question Do you enjoy browsing/shopping clothes?

Explanation *You like browsing clothing items or regularly go shopping for clothes or similar activities.*

Answer Yes | No

A.2 Testing Framework

The data points directly related to the testing framework are split into statements rated on 5-point Likert scales and regular questions and data measures.

Likert Scale Statements

These are statements where participants indicated their rate of approval on a 5-point Likert (discrete) scale. Instead of the actual values text representations were shown, they are mapped as follows:

1 Strongly disagree | 2 Disagree | 3 Neutral | 4 Agree | 5 Strongly agree

Some statements measure the level of disapproval. Here the mapping is reversed (1, strongly agree to 5, strongly disagree).

Perceived Accuracy

- I am confident that the product I selected to "purchase" is really the best choice for me.

Perceived Effort

- I easily found the information I was looking for.
- Looking for a product using this app required too much effort. (*reverse scale*)

Intention to Purchase

- I would purchase the product I just chose if given the opportunity.

Intention to Return

- If I had to search for a product on a mobile device in the future and an app like this was available, I would be very likely to use it.
- I don't like this app, so I would not use it again. (*reverse scale*)

Explanation Benefits

- The explanation text on top was useful to me.
- The explanation text on top made my choice more difficult. (*reverse scale*)

Regular Questions

Objective Accuracy

Before answering this question, participants were shown alternatives with similar attributes to their selected item.

Question Did you choose one of the presented alternatives?

Answer Yes | No, I stuck with my initial choice

Objective Effort

This data points are measured by the application itself. *Not filled out by participant.*

- Enter the displayed task completion time.
- Enter the displayed number of critiquing cycles

Quality Insurance

This additional data helped to verify that mostly different items were chosen and both types of critiquing (positive and negative) were used. *Not filled out by participant.*

- Enter the clothing id you have chosen.

- Enter the number of positive cycles (out of the total above).

System Preference

This question was only asked once after the participant evaluated both variants. Additional comments were written down separately. *Not filled out by participant.*

Question Which variant did you prefer (and why)?

Answer Similarity | Similarity+Diversity

Appendix B

User Modeling Appendix

B.1 Stereotype Scores

Category	Attribute	Indie	Emo	Preppy	Gothic	Urban
Age	Child	2	0	7	0	1
	Teenager	10	10	5	7	7
	Young Adult	8	7	6	8	9
	Adult	6	3	8	6	5
	Older Adult	3	1	7	4	2
	Senior	1	0	4	1	1
Jobs	Pupil	7	8	6	7	5
	Student	10	6	6	7	7
	Manager	2	1	8	0	2
	Salesperson	5	5	9	5	5
	Cashier	5	6	5	6	5
	Cook	4	4	4	4	4
	Waiter	7	6	3	6	6
	Nurse	7	7	3	6	6
	Customer Service	5	5	8	3	5
	Carpenter	4	2	3	5	4
	Secretary	3	2	6	5	3
	Assistant	6	4	6	3	3
	Lawyer	3	1	9	1	2
	Programmer	6	7	2	7	6
	Athlete	2	0	5	1	5
	Researcher	4	4	4	5	4
	Unemployed	5	5	2	6	7
	Other	5	5	5	5	5
Music Taste	Electronic	9	1	2	1	6

Category	Attribute	Indie	Emo	Preppy	Gothic	Urban
	Pop	4	6	4	2	7
	Rock	2	7	4	7	5
	Classic	0	1	6	3	2
	Jazz	0	0	6	1	1
	DnB	4	3	4	2	6
	Hip Hop	4	0	1	1	8
	Folk	7	2	6	2	4
	Indie	9	6	5	3	5
Gender	Male	1	1	1	1	1
	Female	1	1	1	1	1
Brands	Adidas	5	1	5	2	7
	Allegra K	6	6	6	7	2
	Boom Bap	9	4	3	2	8
	Boss	2	1	9	1	2
	Brax	2	1	9	1	2
	Bye Bye Kitty	4	9	1	9	2
	C&A	4	2	5	3	4
	Carhartt	6	1	2	2	8
	Chanel	3	1	9	1	2
	Converse	7	5	2	3	8
	Cupcake Cult	6	9	1	9	2
	DC	4	1	1	1	8
	Denim	4	4	4	2	8
	Dickies	4	2	1	2	9
	Diesel	3	3	5	5	6
	Dior	2	1	9	1	2
	Esprit	6	3	6	3	7
	Etnies	4	3	2	3	8
	Fjällräven	3	3	6	5	5
	Forever 21	9	5	5	5	6
	Gstar	7	3	7	4	7
	Gucci	2	1	9	1	2
	H&M	5	1	6	3	6
	H&R London	2	9	1	9	2
	Hell Bunny	2	9	1	8	2
	Innocent	2	9	1	9	1
	J.crew	4	2	9	1	4
	Lacoste	3	1	9	1	5
	Levis	6	3	7	4	7
	Living Dead Souls	2	8	1	9	2
	Louis Vuitton	2	1	9	1	2
	LRG original	4	2	1	2	9
	Mazine	4	2	1	2	9

Category	Attribute	Indie	Emo	Preppy	Gothic	Urban
	New Yorker	6	2	6	3	7
	Nike	4	2	5	2	7
	Pepe	6	3	5	3	7
	Prada	2	1	9	1	2
	Ralph Lauren	4	1	9	1	2
	Reebok	4	2	5	2	7
	S.Oliver	6	1	7	1	6
	Scotch & Soda	8	3	6	3	7
	Spiral	2	9	1	9	2
	Superdry	5	1	6	4	5
	Tom Tailor	3	2	4	3	5
	Tommy Hilfiger	4	2	6	2	5
	Vans	5	3	3	2	8
	Versace	2	1	9	5	2
Item Attributes	Acryl	5	2	3	1	2
	Athletic	3	2	3	1	6
	Baby Blue	6	2	7	2	5
	Baggy	2	1	1	1	7
	Black	4	8	4	9	6
	Blue	5	4	5	5	6
	Bow	3	7	1	7	2
	Bright	6	2	7	2	5
	Brown	5	6	4	5	6
	Classic	1	1	6	4	2
	Cords	1	5	1	8	1
	Cream	6	4	5	5	5
	Dark	5	9	4	9	5
	Emo	2	10	1	5	1
	Girly	2	5	4	3	2
	Green	5	3	5	4	4
	Grey	5	7	5	6	5
	Hoody	5	5	2	2	8
	Leather	2	3	5	8	1
	Logo	7	5	2	1	7
	Navy	4	3	6	2	6
	Neon	5	7	4	2	6
	Original	10	5	1	5	7
	Pattern	5	5	2	5	5
	Pink	5	6	2	1	5
	Plush	2	5	1	1	1
	Purple	5	6	5	6	4
	Rectangle	4	5	1	6	4
	Red	5	7	5	7	4

Category	Attribute	Indie	Emo	Preppy	Gothic	Urban
	Retro	8	5	2	2	7
	Romantic	1	7	1	8	2
	Short	5	5	4	5	5
	Slogan	7	3	2	2	7
	Sport	4	2	6	2	6
	Sporty	4	2	6	2	6
	Street	4	1	1	1	9
	Stripes	5	4	1	5	5
	Tight	8	8	2	5	5
	Used	9	2	1	1	8
	Vintage	8	5	2	1	8
	White	6	4	6	5	5
	Yellow	4	3	6	2	5

Table B.1: Weighted attributes for stereotypes Indie, Emo, Preppy, Gothic and Urban

Category	Attribute	Athlete	Skater	Girly	Mainstream	Classy
Age	Child	1	2	8	2	0
	Teenager	8	8	8	3	0
	Young Adult	7	6	7	4	3
	Adult	5	3	4	7	4
	Older Adult	3	1	2	7	7
	Senior	1	0	0	5	9
Jobs	Pupil	6	8	7	4	0
	Student	7	7	6	5	2
	Manager	3	2	0	7	4
	Salesperson	5	3	2	4	3
	Cashier	3	3	3	5	2
	Cook	4	4	2	5	3
	Waiter	4	4	3	4	4
	Nurse	3	3	5	4	3
	Customer Service	4	3	3	5	3
	Carpenter	7	2	0	5	2
	Secretary	4	2	3	7	6
	Assistant	5	2	5	7	6
	Lawyer	2	1	1	8	8
	Programmer	4	6	1	4	3
	Athlete	10	4	1	6	2
	Researcher	3	2	2	7	7

Category	Attribute	Athlete	Skater	Girly	Mainstream	Classy
	Unemployed	5	6	3	6	2
	Other	5	5	5	5	5
Music Taste	Electronic	3	3	4	4	0
	Pop	5	2	6	8	2
	Rock	6	6	2	7	2
	Classic	2	0	0	3	8
	Jazz	1	0	0	2	8
	DnB	5	6	1	2	1
	Hip Hop	6	7	1	2	1
	Folk	3	3	6	3	6
	Indie	3	2	7	4	2
Gender	Male	1	1	0	1	1
	Female	1	1	1	1	1
Brands	Adidas	9	7	4	7	2
	Allegra K	2	2	9	3	4
	Boom Bap	5	7	6	4	2
	Boss	5	1	2	6	9
	Brax	5	1	2	7	9
	Bye Bye Kitty	2	2	4	3	2
	C&A	5	3	5	9	5
	Carhartt	6	7	4	5	2
	Chanel	2	2	5	5	8
	Converse	7	8	3	5	1
	Cupcake Cult	2	2	6	4	1
	DC	7	9	2	4	1
	Denim	7	7	4	6	2
	Dickies	7	9	3	5	1
	Diesel	7	5	2	8	4
	Dior	2	1	5	5	8
	Esprit	6	6	5	8	5
	Etnies	7	9	3	5	1
	Fjällräven	8	5	4	7	3
	Forever 21	4	3	8	6	3
	Gstar	6	5	5	8	5
	Gucci	3	1	6	5	8
	H&M	5	4	6	8	4
	H&R London	2	1	2	2	2
	Hell Bunny	3	1	4	2	1
	Innocent	3	1	2	1	1
	J.crew	4	2	2	7	9
	Lacoste	5	2	3	6	9
	Levis	6	5	2	8	5
	Living Dead Souls	2	1	2	1	1

Category	Attribute	Athlete	Skater	Girly	Mainstream	Classy
	Louis Vuitton	2	1	5	5	9
	LRG original	5	7	3	4	1
	Mazine	6	8	2	5	1
	New Yorker	6	6	6	8	3
	Nike	9	5	4	7	2
	Pepe	6	5	4	8	5
	Prada	2	1	6	5	9
	Ralph Lauren	4	1	5	6	8
	Reebok	9	6	3	6	2
	S.Oliver	6	5	5	8	6
	Scotch & Soda	6	5	6	8	4
	Spiral	2	1	2	1	1
	Superdry	5	3	3	7	5
	Tom Tailor	5	4	3	8	5
	Tommy Hilfiger	5	5	3	8	6
	Vans	6	9	2	5	2
	Versace	2	1	5	4	9
Item Attributes	Acryl	5	2	3	1	2
	Athletic	3	2	3	1	6
	Baby Blue	5	5	7	7	8
	Baggy	2	1	1	1	7
	Black	6	5	4	7	8
	Blue	5	4	5	5	6
	Bow	1	1	7	2	3
	Bright	5	5	7	7	6
	Brown	5	6	4	5	6
	Classic	2	1	2	4	9
	Cords	1	1	7	2	3
	Cream	6	4	5	5	5
	Dark	5	9	4	9	5
	Emo	1	1	3	2	1
	Girly	2	3	9	4	1
	Green	5	5	5	6	3
	Grey	5	5	3	6	6
	Hoody	7	7	2	4	1
	Leather	1	1	1	2	3
	Logo	7	7	3	6	1
	Navy	6	5	4	7	6
	Neon	4	4	7	3	2
	Original	3	6	6	2	1
	Pattern	6	6	6	5	2
	Pink	4	3	9	3	3
	Plush	2	1	7	1	1

Category	Attribute	Athlete	Skater	Girly	Mainstream	Classy
	Purple	5	5	6	5	4
	Rectangle	4	5	1	6	4
	Red	4	4	6	5	5
	Retro	2	3	3	4	5
	Romantic	1	1	5	2	3
	Short	7	5	5	4	3
	Slogan	7	7	4	5	1
	Sport	9	6	2	4	2
	Sporty	9	6	2	4	2
	Street	5	8	1	6	1
	Stripes	5	5	6	5	3
	Tight	5	6	5	6	5
	Used	5	8	3	6	1
	Vintage	8	5	2	1	8
	White	6	4	7	5	8
	Yellow	5	5	6	5	5

Table B.2: Weighted attributes for stereotypes Athlete, Skater, Girly, Mainstream and Classy

The age groups are: *Child* = less than 13 years old, *Teenager* = 13-18, *Young Adult* = 18-30, *Adult* = 30-50, *Older Adult* = 50-65, *Senior* = older than 65 years.

Clothing item attributes and brands were selected upon frequently occuring in the overall dataset.

B.2 Stereotype User Study Survey

The following is a reflection of the questionnaire given to participants. It is not the original, but questions and possibilities for answers are identical. As all participants were German, the language of the questionnaire was German as well. It is hereby translated into English.

General Questions

After you have tested the application in both runs, we would like to ask you to provide some information about yourself and give us some feedback concerning the application.

Question If you would like to participate in the lottery, please provide your e-mail address. You can win an iPod Shuffle.

Answer Text field

Question Which user id has been given to you by the system

Answer Text field

Question Do you own a smartphone?

Answer Yes | No

Question If you own a smartphone, do you use mobile applications on it?

Answer No, never. | Yes, rarely. | Yes, habitually. | Yes, consistently.

Question How would you describe your clothing style?

Answer Athletic | Emo | Girly | Gothic | Indie/Hipster | Preppy | Skater | Urban | Mainstream | Classy | Other: [*Text field*]

Likert Scale Statements

In this section, participants are asked to rate the following statements on a 5-point Likert (discrete) scale.

1 Strongly disagree | 2 Disagree | 3 Neutral | 4 Agree | 5 Strongly agree

- I am interested in the use of a mobile application which supports me in finding suitable clothing items in nearby stores.
- I would use the application I just tested to support me in finding clothing items in nearby stores. (If not, why?)
- I liked the design of the application.
- I understood quickly how to use the application.
- I am satisfied with the item I last selected and would like to take a closer look at it.

- I was satisfied with the number of clothing items available.
- I found the form at the beginning of the application disturbing.

Questions Concerning the Application

Question The following things confused me while using the app:

Answer Text field

Question In which test run did the clothing items reflect your clothing style the best?

Answer Run 1 | Run 2 | None of the two

Question How many questions do you consider to be acceptable to answer before being able to use the app?

Answer 0 | 1-2 | 3-6 | 6-10 | 11-20

Question Did you use the filters to narrow down the visible items?

Answer Yes | No

Question Did you use more than one filter at the same time?

AnswerYes | No

Question How often did you scroll until you found an item of choice?

Answer Never | Once | Several Times | Until I found an item

Question What did you like about the application?

Answer Text field

Question What did you not like about the application?

Answer Text field

Appendix C

Interaction Design Appendix

C.1 Online Survey

Participants of the online survey first received a brief explanation of their task. The online survey also displayed the different hand-sketched paper prototypes for each interface design, so that the participants could imagine their functionality. Afterwards, the users were asked to fill in a form containing demographic questions, as well as questions regarding the different interface designs.

Explanation of the Study

This form is to find out how a human can imagine its interaction with a mobile shopping recommender system. A mobile shopping recommender system should help you in finding the best product you want to purchase. From Wikipedia: "Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item or the user's social environment." The word 'mobile' means, this system would be available as an application on your smartphone and you would be able to use it anytime/anywhere. Imagine a scenario where you want to buy a shirt, for example. You have a very specific shirt in your mind. The process of choosing the best fitting shirt for you follows in three steps: First, you set your initial preferences about that shirt, the color, size, price, but also contextual preferences, your location, is the shop currently opened, does it have an online shop, and so on. Second, the system proposes a set of items regarding your initial preferences. Either you choose an item you like, or you decide to critique an item and tell the system what you like/don't like about it and expect better results. Finally, after several cycles of critiques, you choose an item you like most or you cancel the process. This survey is to find out how you want to interact with the system

in order to make this process as usable and easy as possible, keeping you satisfied with it. Several questions will be asked about your preferences. The three steps of the process present three main views that have to be graded: How to set the initial preferences of the item you are looking for; how the resulting set of recommended items is being presented; and how to give feedback about a recommended item.

Demographic Questions

Question How old are you?

Answer Text field

Question Which is your gender?

Answer Male | Female

Question What is your current profession?

Explanation *e.g., student, teacher, account manager*

Answer Text field

Question How often do you use a recommender system for shopping?

Answer Never | Rarely | Occasionally | Often | Very often

Question If the answer to the previous question is not "never", which recommender system did/do you use?

Answer Text field

Question If the answer to the previous question is not "never", which recommender system did/do you use?

Explanation *e.g., Amazon, idealo*

Answer Text field

Question How often do you use a smartphone?

Answer Never | Rarely | Occasionally | Often | Very often

Question Have you made experiences with using a mobile recommender system?

Answer Yes | No

Question If yes, which one?

Answer Text field

Questions Regarding the Setting of Initial Preferences

Question I would like to set the initial preferences of the item by taking a picture of the item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to set the initial preferences of the item by uploading a picture of the item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to set the initial preferences of the item by setting preferences one by one manually.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to choose from a predefined set of values in order to set the initial preferences of the item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to define my own preferences for an item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to set the initial preferences of the item by being asked questions by the system.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question The maximum number of question cycles I find acceptable is:

Answer 1-3 | 4-6 | 7-9 | 10+

Question I would like the system to choose items implicitly.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question From the proposed five ways of collecting preferences, my favorite three options are:

Answer Taking a picture | Uploading a picture | Setting preferences one by one manually | Answering questions | Implicitly

Question I can also imagine setting preferences by:

Answer Text field

Question I find the following item preferences as a must for a mobile shopping recommender

Answer Size | Price | Style | Color | Designer | Fabric | Other: *[Text field]*

Question I find the following context preferences as a must for a mobile shopping recommender

Answer Location | Weather | Timestamp | Social network friends | Opening hours of a shop | Availability of online shop | Traveling costs | Presence of motion | Other: *[Text field]*

Questions Regarding the Presentation Interfaces

Question I would like to be able to set the sorting criteria of the items:

Answer At the beginning once | At any time | Other: *[Text field]*

Question I would like to be presented only one item with a detailed explanation.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to be presented multiple items in a list.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to be presented multiple items in a grid.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to be presented multiple items on a map.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question My favorite proposed way of presenting recommendations is:

Answer Single item | Multiple items in a list | Multiple items in a grid | Map

Question My least proposed way of presenting recommendations is:

Answer Single item | Multiple items in a list | Multiple items in a grid | Map

Question I find it important to be given an additional search field.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I find it important to be given a "see more recommendations" button.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I find it important to be able to modify my initial preferences at any time.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I find it important to be able to compare two items.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I find it important to be able to exclude an item from the resulting set.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I find it important to be able to see explanations why an item is in the resulting set.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question Do you find anything else important?

Answer Text field

Questions Regarding the Feedback Strategy

Question I would like to see alternatives the system proposes me for a recommended item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question I would like to manually adjust an item.

Answer Strongly disagree | Disagree | Neutral | Agree | Strongly agree

Question Giving feedback for a preference is best with:

Answer A like/dislike option | Rating stars | Comparison | Positive/Negative critiques | Other: *[Text field]*

Question Is there any other way of giving feedback you can imagine in a mobile shopping recommender system?

Answer Text field

C.2 Semantic Clothing Attributes

Attribute	Values
Gender	Men, Women
Men Items	"Any","Cardigan", "Coat", "Gilet", "Hoodie", "Jacket", "Jeans", "Jumper", "Pyjama", "Shirt", "Shorts", "Suit","Sweater", "Swim Shorts", "T-Shirt", "Top", "Trousers",
Women Items	"Any","Bikini", "Blazer","Blouse", "Bolero", "Cardigan", "Coat", "Dress", "Gilet", "Jacket", "Jeans", "Jumper", "Parka", "Pyjama", "Shirt","Skirt", "Suit","Sweater", "Swimsuit", "T-Shirt", "Top", "Trousers","Tunic"
Price	Numerical Range (from x to y)
Style	"Any", "Alternative","Business", "Casual", "Clubwear", "Elegant","Formal","Summer/Beach","Sporty", "Trendy"
Brand	"Any","7 for all Mankind","Adidas","Armani", "Bershka","Benetton", "BOSS", "C&A", "Desigual", "Diesel","Esprit","even&odd", "Gant","Guess","H&M","Levi's","Mango", "Marc O'Polo", "Massimo Dutti","Mexx", "Michael Kors", "Nike", "ONLY","Pepe Jeans", "Puma","Ralph Lauren", "Replay", "S'Oliver", "Stefanel", "Stradivarius", "Tom Tailor", "Tommy Hilfiger", "Versace", "Zara"
Color	"Any","Beige","Black","Blue","Brown","Cyan", "Green", "Gold", "Orange", "Pink", "Purple","Red","Silver","Yellow", "White"
Size	"Any","XS","S","M", "L", "XL", "XXL"
Fabric	"Any", "Angora", "Cotton", "Elastane", "Fur", "Leather", "Polyester", "Silk", "Synthetics", "Viscose", "Wool"
Pattern	"Any", "Animal-Printed", "Floral", "Graphics", "Plaid", "Spotted", "Striped", "Unicolor"
Sleeve	"Any","3/4-sleeve", "Long sleeve", "No sleeve","Short sleeve"
Length	"Any","Calf length","Extra short","Full length", "Knee length","Short/Mini"

C.3 Usability Questionnaires in the User Study

Each of the questions (except for question eighteen, which could be answered in a text field) are answered on a 7-point Likert (discrete) scale. Instead of the actual values, text representations were shown. They are mapped as follows:

1 Strongly disagree | 2 Mostly disagree | 1 Somewhat disagree | 4 Neutral | 5 Somewhat agree | 6 Mostly agree | 7 Strongly agree

Likert Scale Statements Regarding the Setting of Initial Preferences

- It was simple to use this interface.
- It was easy to use this interface.
- I became familiar with the interface very quickly.
- This is an adequate way for me to express my preferences.
- The interface provides an adequate way for me to revise my preferences.
- I found it easy to tell the system about my preferences with this interface.
- I feel in control of telling the recommender system what I want with this interface.
- I feel I could use this interface in a quick, productive way.
- I am all in all satisfied with this interface.
- I think that most people would learn very quickly to deal with this interface.
- I think that most people could work with this interface for a long time without input errors.

Likert Scale Statements Regarding the Presentation Interfaces

- It was simple to use this interface.
- It was easy to use this interface.
- I became familiar with the interface very quickly.
- The interface provides sufficient information.
- The information provided for the recommended items is sufficient for me.
- The labels of the recommender interface are clear and adequate.
- The layout of the recommender interface is attractive and adequate.
- Looking for a recommended item required too much effort (reverse scale).
- The interface helped me understand why the items were recommended to me.
- An additional search field would have helped me in finding the item best suited.
- An additional sort option would have helped me in finding the item best suited.
- I wish I had been able to exclude some recommended items.
- I wish I had been able to compare two recommended items.

- I think that most people would learn very quickly to deal with this interface.
- I think that most people could work with this interface for a long time without input errors.

Likert Scale Statements Regarding the Feedback Strategy

- It was simple to use this interface.
- It was easy to use this interface.
- I became familiar with the interface very quickly.
- The interface is an adequate way for me to revise my preferences.
- It is easy for me to inform the system if I dislike/like the recommended item.
- The layout of the interface is attractive and adequate.
- I feel I could use this interface in a quickly, productive way.
- I am all in all satisfied with this interface.
- I think that most people would learn very quickly to deal with this interface.
- I think that most people could work with this interface for a long time without input errors.

C.4 Demographics and Shopping Experience Questionnaire

Questions Regarding the Shopping Experience

Question What is your name?

Answer Text field

Question How old are you?

Answer Text field

Question Which is your gender?

Answer Male | Female

Question What is your current occupation?

Answer Student | Employee | Researcher | Self-employee | Other: [*Text field*]

Question Do you enjoy browsing/buying clothes?

Answer Yes | No

Question Are you familiar with the term "recommended item"?

Explanation *e.g., when you search for a product on Amazon, Amazon recommends you items based on your browsing history or what other customers with similar preferences to yours have bought.*

Answer Yes | No

Question Have you ever bought an item that was recommended to you on a shopping site?

Answer Yes | No

Question Do/Did you have a smartphone?

Answer Yes | No

Question How often did/do you use your smartphone?

Answer Never (1) | Rarely (2) | Occasionally (3) | Often (4) | Very often (5)

Question Do/Did you shop from your smartphone?

Answer Yes | No

Appendix D

Context-Awareness Appendix

D.1 User Preferences for Categories of Clothes

In section 7.2.1 the relevance of contextual factors was measured by the normalized mutual information between the influence stated by the user and each contextual factor: The higher the mutual information, the more likely a specific contextual factor influences the user's purchasing decision. In the following table, we present an overview of the contextual factors ordered by different clothing categories:

Table D.1: Measured relevance

Tops		**Dresses**		**Underwear**		**Cardigans**	
day of the week	0.85	time of the day	1	time available	0.92	day of the week	1
temperature	0.84	day of the week	1	day of the week	0.9	weather	0.92
time available	0.84	weather	1	time of the day	0.9	temperature	0.92
transport	0.81	time available	1	crowdedness	0.88	time available	0.87
weather	0.8	budget	0.93	season	0.83	mood	0.87
time of the day	0.8	companion	0.91	budget	0.81	crowdedness	0.82
crowdedness	0.78	temperature	0.91	transport	0.81	companion	0.82
intent of purchase	0.78	season	0.88	temperature	0.81	intent of purchase	0.82
companion	0.76	transport	0.78	weather	0.79	budget	0.81
season	0.76	intent of purchase	0.76	companion	0.78	time of the day	0.79
budget	0.76	crowdedness	0.74	mood	0.75	season	0.73
mood	0.71	mood	0.74	intent of purchase	0.69	transport	0.73
Trousers		**Coats**		**Blouses**		**Jackets**	
intent of purchase	1	temperature	1	time of the day	1	budget	0.92
weather	1	time available	0.91	day of the week	1	companion	0.9
day of the week	0.89	transport	0.89	transport	1	day of the week	0.89
time available	0.87	budget	0.87	time available	0.9	transport	0.89

Continued on next page

Table D.1 – *Continued from previous page*

transport	0.84	day of the week	0.86	intent of purchase	0.89	temperature	0.88
mood	0.83	mood	0.85	mood	0.89	time available	0.87
companion	0.82	crowdedness	0.82	weather	0.85	intent of purchase	0.81
budget	0.81	season	0.81	temperature	0.82	crowdedness	0.8
temperature	0.8	intent of purchase	0.8	budget	0.82	time of the day	0.79
crowdedness	0.8	time of the day	0.76	crowdedness	0.8	season	0.77
season	0.74	weather	0.75	companion	0.75	mood	0.77
time of the day	0.74	companion	0.68	season	0.74	weather	0.74
Skirts		**Jeans**		**Socks**		**Swimwear**	
time available	1	time available	1	mood	1	budget	1
budget	1	companion	0.91	season	1	temperature	0.92
day of the week	1	temperature	0.9	crowdedness	0.92	day of the week	0.9
crowdedness	1	day of the week	0.87	time available	0.89	time available	0.88
companion	1	time of the day	0.86	intent of purchase	0.88	crowdedness	0.84
intent of purchase	0.88	crowdedness	0.83	day of the week	0.86	weather	0.82
temperature	0.88	transport	0.82	budget	0.84	intent of purchase	0.8
weather	0.87	budget	0.81	temperature	0.8	mood	0.8
mood	0.86	season	0.81	companion	0.79	season	0.8
season	0.85	mood	0.77	time of the day	0.77	transport	0.79
time of the day	0.77	weather	0.74	weather	0.73	companion	0.76
transport	0.74	intent of purchase	0.72	transport	0.68	time of the day	0.75
Suits		**Shirts**					
budget	1	budget	1				
intent of purchase	1	time of the day	1				
companion	1	day of the week	1				
season	1	intent of purchase	1				
time of the day	1	weather	1				
time available	1	season	1				
weather	0.92	temperature	1				
temperature	0.89	time available	1				
crowdedness	0.84	companion	0.9				
mood	0.83	crowdedness	0.87				
day of the week	0.82	mood	0.85				
transport	0.72	transport	0.85				

D.2 Context Scenarios

We used five different context scenarios in the user study:

- Imagine that you want to buy clothes for daily wear, you are a budget buyer and the temperature is cold. You don't care about the distance to the shops.
- Imagine that you want to buy clothes for daily wear, you are a budget buyer and the temperature is hot. You want to get suggestions of stores that are nearby (within a radius of 2km).
- Imagine that you are feeling like a party animal, it is weekend and you are a budget buyer. You don't care about the distance to the shops.
- Imagine that you want to buy clothes for working purposes, you are willing to spend a high amount of money and the temperature is warm. You want to get suggestions of stores that are nearby (within a radius of 2km).
- Imagine that you want to buy clothes for sports purposes, you are feeling outdoorsy and the temperature is hot. You want to get suggestions of stores that are nearby (within a radius of 2km).

D.3 User Study Survey

Demographic Questions

The survey starts with some demographic questions.

Question What is your primary profession?

Answer Text field

Question How old are you?

Answer Text field

The Post-Study Questionnaire

The participants are asked to fill out the questionnaire after they have completed all the scenarios in the user study.

Likert Scale Statements

In this section, participants are asked to rate the following statements on a 5-point Likert (discrete) scale.

1 Strongly disagree | 2 Disagree | 3 Neutral | 4 Agree | 5 Strongly agree

Context-Aware System:

- It was easy to find the information I needed.
- The system delivered accurate recommendation in order to complete the scenario.
- I like using this system.
- I understood the benefit of using the contextual conditions.
- I am satisfied with the provided contextual explanations.
- I believe that the contextual explanations are useful.
- The contextual explanations provided by this system are clear.

Non-Context-Aware System:

- It was easy to find the information I needed.
- The system delivers accurate recommendation in order to help me to complete the scenario.
- I like using this system.

Regular Questions

In this section, participants need to choose between two candidate options: The system she has tested first (system 1) or last (system 2).

Question Which system do you prefer?

Answer System 1 | System 2

Question Which system suggests more appropriate clothes?

Answer System 1 | System 2

Appendix E

Explanations Appendix

E.1 User Study Survey

Demographic Questions

The survey starts with some demographic and background related questions to allow statements about the sampled population.

Question How old are you?

Answer Text field

Question What gender are you?

Answer Male | Female

Question What is your primary profession?

Answer Student | Employee | Researcher | Self-Employed | Other: [*Text field*]

Question How often do you use a smartphone?

Answer Never | Rarely | Occasionally | Often | Very often/Always

Question Do you have online shopping experience?

Explanation *Browsing and ordering from Amazon, eBay, hardwareversand or other online shops.*

Answer Yes | No

Question Do you shop from your mobile device?

Explanation *Through an application or the web browser on your phone, tablet or a similar device.*

Answer Yes | No

Question Do you enjoy browsing / shopping clothes?

Explanation *You like browsing clothing items or regularly go shopping for clothes or similar activities.*

Answer Yes | No

Testing Framework

The data points directly related to the testing framework are split into statements rated on 5-point Likert scales and regular questions.

Likert Scale Statements

These are statements where participants indicated their rate of approval on a 5-point Likert (discrete) scale. Instead of the actual values, text representations were shown. They are mapped as follows:

1 Strongly disagree | 2 Disagree | 3 Neutral | 4 Agree | 5 Strongly agree

Some statements measure the level of disapproval. Here the mapping is reversed (1, strongly agree to 5, strongly disagree).

- I understand what the system bases its recommendations on.
- I don't feel in control of telling the system what I want. *reverse scale*

- I find it hard to correct any wrong assumptions made by the recommender system. *reverse scale*
- Looking for a product using this application required too much effort. *reverse scale*
- Overall, I am satisfied with the recommender system.

Regular Questions

- Please explain how you think the system calculates the recommendations.
- Do you have any further remarks regarding this system?

These questions were only asked once after the participant evaluated both variants. Additional comments were written down separately. *Not filled out by the participant.*

- Which variant did you prefer the most?
- Why did you prefer that variant? (In case you preferred neither, then why?)

Appendix F

Final Evaluation Appendix

F.1 Eliciting Context Factors for Context-Aware Recommendation

Before being able to recommend items based on the current context, the relevant context has to be defined. For the assessment the different context factors are presented in more detail and classified into the different aspects of context (physical, social, modal and interaction media). It is argued why or why not they should be part of the prototype. The *physical context* describes the immediate physical environment of the user and the environment's states. The following context factors were identified for the physical context:

Location defines the place from which the user accesses the application. Location is normally determined by reading GPS data in conjunction with data of network cells of the mobile communication system operator. From the location, data like the distance to a shop can be inferred and the map of shops in the surroundings is created. As this context factor is relatively easy to elicit and is expected to be available from a lot of users, it is part of this implementation.

Temperature defines the current environment temperature in Celsius. The temperature can be estimated by sending a request to a logical sensor, such as OpenWeatherMap [ExtremeElectronics, 2015] or Yahoo! Weather [Yahoo, 2015]. To fulfill the request, the weather API (Application Programming Interface) requires the coordinates or the name of a place nearby to send accurate weather data. Although this weather data is not always precise and might differ from the real temperature, it is a good estimation. The temperature data could influence whether the user is likely to buy clothes as t-shirts, shorts or swimwear (in case of warm weather). As it is expected that this factor has a significant influence it is integrated into the prototype. In our final prototype the temperature is represented in the following groups: below 0° C, above 30° C and in five degree intervals from 0 to 30.

Weather describes the current conditions like rain or sunshine. It is measured via the same APIs as temperature and to some degree correlates with it, e.g., snow is only possible when it is very cold. Weather data could be relevant to detect if the user was searching for a jacket (in case of rain) or swimwear (in case of sunshine). Weather data is also integrated into the prototype and can take one of the following conditions: Sunny, partly cloudy, cloudy, mostly cloudy, raining or snowing.

Wind is available from weather APIs like temperature. However, it is not expected that wind is able to influence the decision on which item a user wants to buy. It could be possible, that it alters the decision between a wind-proof jacket and a coat, but we expect this to be unlikely and therefore to not integrate wind into the recommender system.

Brightness describes how bright it is at this given day. One could argue that the brighter the day is, the happier the people are and the more likely they are to buy clothes. Brightness is hard to measure as it is usually not available from weather APIs. The device's camera could be used to detect the brightness in the current user's position. However, brightness is not integrated into the prototype, as it is relatively hard to measure and not expected to provide significant benefits.

Noise is usually measured in decibel and could be measured by the device's microphone. The noise level could provide information on how stressed the user might be due to her environment and what kind of environment she is in. However, it is not clear how the noise level could influence shopping behavior. Therefore, noise is not integrated into the prototype.

Season can take one of the values fall, winter, spring and summer. It can be inferred from the current date. In general the shops already only offer fashion items that fit the current season. Therefore, there is no reason to integrate this parameter into the prototype as only season specific items are available. Additionally, it can be partly substituted by the weather and temperature context factors, which are part of the prototype.

Day of the week is used to distinguish working days and holidays (including weekends and public holiday). This can directly be derived from the current date. However, as there are some jobs in which people work on holidays (e.g., restaurants), it might be necessary to adapt this scheme for those people. We expect that on holidays people search for more extravagant clothes than on working days. Therefore, this context factor is integrated into the prototype.

Time of the day can also be derived from the current device's time. It could be possible that certain item's are bought more often in the evening (when it is dark outside) than during midday, e.g., items with reflectors. Furthermore this context factor is necessary to detect whether a shop is open at a given time. Therefore, time of the day is integrated into the prototype. In the implementation of the final prototype we distinguish between night, morning, midday, afternoon and evening.

Activity describes what the user is currently doing like walking, running, driving a car, taking the train or riding a bicycle. Of course it is interesting for distance calculations to determine how fast a user moves. It therefore should give an indication of how many shops and in which distance they should be integrated into the recommendations. But it is not expected, that it changes the way a user perceives the recommendations. Activity recognition still is a current research topic [De Pessemier et al., 2013], as each activity has to be learned user-specific. Therefore, we do not integrate activity into the prototype.

Opening hours of the shop are used to determine whether it is possible for a user to reach a given shop within its operating hours. This is especially useful in the evening or the morning, when the shops might not be open. As this context factor could also be used to reduce the recommender system's workload and is useful for the users, it is integrated into the prototype.

Item is in stock describes whether an item is currently available. It seems straightforward to only take items into account, that are in stock. However, it is possible that users want to be inspired by items, that are not available anymore, as different shops usually provide clothing items for a specific clothing style. Therefore, the user might find items, which are not available anymore. Nevertheless, she still wants to go to the shop as she expects to find similar items there. This context factor can be used as a pre-filtering criterion to reduce the recommender systems workload and is therefore integrated into the prototype. However, the user can decide at the beginning of the recommendation process, if the algorithm should take this context factor into account.

Crowdedness describes how many people might be within a shop and therefore implicitly determines how popular a shop is. This data could be available by measuring how often items of a shop are selected and then it is determined statistically how crowded the shop might be. For some people it is important that a shop is not too crowded, because they do not like to be surrounded by too many people or do not want to wait long at the cashier's desk. As this context factor can also be used to reduce the number of shops from which items can be selected, the recommender system's workload is reduced. Therefore, we integrated this context factor into the prototype. Again, the user can state at the beginning of the recommendation process whether this context factor should be taken into account or not.

The *social context* can be defined as the presence of other people around the user, or the influence of these users regarding the current task. For social context there are the following context factors defined:

Company defines who accompanies the user when items are recommended. This can be either friends, the family, or the user is on her own. We cannot directly measure this factor, unless other application users are willing to share their current location as well. The system could then detect social relationships, if the user added other

persons as friends or family. However, the user could also input this information at the beginning of the recommendation session. This context factor will be considered by the prototype. Research has shown that whether you are shopping with friends or family does have an influence on your shopping behavior [Luo, 2005]. Therefore, the company context factor can be: Alone, with a friend or with the family.

People around is an abstract measure of how crowded the place around the user is. The system could try to detect other devices via Bluetooth or the wireless network to have an indication of how crowded the place is. A person nearby might influence the shopping behavior, e.g., a user might purchase different products when feeling observed by others. Argo et al. tested this hypothesis in a real shopping environment, but as the mobile device is a more private environment, we do not expected that the results of Argo et al. hold in this environment and therefore do not integrate this context factor into the prototype [Argo et al., 2005]. Nevertheless, this hypothesis has to be verified.

Network of friends should describe with how many people a person usually corresponds and how large her network of friends is. This could be measured by using the Graph API by Facebook [Facebook, 2015] or other social networks. However, it is unclear if this even influences the user's shopping behavior. Furthermore, it requires a lot of effort to get this data from all the different sources (e.g., by using mobile phone protocols, social networks) and is difficult to use, as the system on which the prototype is evaluated stays the same in our user study. This context factor is not considered, mainly because this data is too difficult to elicit within the user study.

Popular items are items that are frequently selected. This is a context factor as some users prefer to wear what everyone else wears. Sometimes this context factor is already taken into account by classic recommender systems, but as our content-based recommender system does not use such a feature it is considered as an indirect context factor.

Expert opinion is the opinion of an *expert* on whether the item is suitable for the user. An expert can be a blogger or a person in the social environment. It could also be the *wisdom of the crowd* (favoring popular items). However, as every user is different, it is unlikely that selections made by experts match each user's taste. We do not include expert opinion into our recommendation algorithm as the distinction to popular items is difficult and these are already considered. However, we highlight those recommended items that have also been selected by fashion experts as we think that this might positively affect the user experience (see *subsection 9.2.5*).

The *modal context* represents the user's current state of mind. In general, it describes what exactly the user is searching for and is modeled by the content-based recommender system. The content-based recommender system reacts on the expressed user preferences and therefore indirectly models the state of mind. The following context factors could be selected for the mobile shopping scenario:

Intent of purchase describes more exactly what the user is searching for. These might be present for other people or clothing items for a specific task (e.g., a party). This could be easily detectable by asking the user and be used to distinguish between short and long-term preferences. Nevertheless, this context factor is not used in the prototype as user profiles are not saved for future references and in the user survey the participants are expected to behave as if they are searching for something for themselves.

Mood in general describes how the user feels. However, in other works, such as [Savage et al., 2012] it is more generally used as a description for different categories of items (similar to intent of purchase). The *real* mood is hard to elicit from the user. It could be measured by the facial expressions or bodily functions of a user. However, it is hard to implicitly measure these factors and even harder to try to imagine a feeling as would be required in the user study. Therefore, this context factor is excluded from the prototype.

Distraction measures the user's attention span. It should give an indication of whether the user is able to focus on the mobile device. This could be inferred, e.g., by noise, brightness, how close the user's face is to the smartphone and how fast she reacts. However, it is unclear how the distraction should influence the application. Therefore, the context factor distraction is not used in the prototype.

Budget describes how much the user is willing to spend for her purchase(s). It can be interpreted as an upper bound for the price. However, it might be possible, that the user wants to buy something impulsively, due to the fact, that she is with friends or because a very appealing product costs more. As this factor is highly dependent on the results of the recommendation process, it is not incorporated into the context, but part of the content-based recommender system, which recommends item based on given price ranges. It is expected that this factor can also change between different clothing types and can therefore not be assumed to be static. As the user usually has a certain budget in mind this context factor is part of the modal context.

The *interaction media* context describes the user's device and its features. The following context factors are part of the interaction media context:

Device is the user's device with which the application is accessed. The device's parameters can be read via built-in functions. They can give an indication of which technical features a user prioritizes, e.g., quality, functionality or size. However, the device is not part of the context, as the device will stay the same for all users throughout the user study.

Installed Apps can give an indication of the user's preferences. Which applications are available on the smartphone is extractable based on built-in functions. Again, this context factor is not used as the device stays the same throughout the user study and therefore no differences can be analyzed. Furthermore, it is not clear how an installed application influences the user's choice of items.

Hence, the final prototype integrated the following context factors: *Time of the day, day of the week, temperature, weather, company, distance to shop, crowdedness, shop opening hours* and *item is in stock*.

F.2 Context Factor Weights Based on Clothing Type

In order to calculate the relevance (weights) of different context factors for an item, weights were assigned to the context factors. These weights depend on the item's clothing type. The weight is used to calculate the contextual distance of the current context to the item. *Subsection 7.2.1* describes how these weights were calculated. Whenever the clothing type was not already tested, the weights were determined by an average of other context factors. These weights are:

Chino is set to the value of Trousers.

Hoodie is set to $0.8 \cdot Cardigan + 0.2 \cdot Jacket$.

Jumper is set to $0.8 \cdot Sweatshirt + 0.2 \cdot Shirt$.

Shorts is set to $0.5 \cdot Swimwear + 0.5 \cdot Trousers$.

Sweatshirt is set to the value of Cardigan.

T-Shirt is set to the value of Shirt.

Tunic is set to $0.5 \cdot Dress + 0.5 \cdot Shirt$.

The resulting weights can be seen in *table F.1*. How these weights are used to calculate distances between the current context and the item is described in *subsection 9.2.4*.

Context factor	Company	Day of the week	Temperature	Time of the week	Weather
Chino	0.82	0.89	0.8	0.74	1.0
Hoodie	0.836	0.978	0.912	0.79	0.884
Jumper	0.836	1.0	0.936	0.832	0.936
Shorts	0.79	1.0	0.955	0.875	0.91
Sweatshirt	0.82	1.0	0.92	0.79	0.92
T-Shirt	0.9	1.0	1.0	1.0	1.0
Tunic	0.905	1.0	0.955	1.0	1.0

Table F.1: Context factor weights based on clothing type

F.3 Distances of Context Factors

The context factors company and weather do not use an euclidean distance function. Instead distances are defined between the different context conditions, which are then used in the contextual post-filtering stage. In *figure F.1* the distance graph for company is defined.

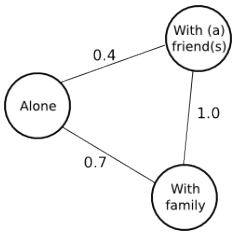


Figure F.1: Distance graph for company

There are too many different context factors for weather, as that they can be adequately presented in a picture of the graph. Therefore, *table F.2* presents the distances between the context conditions for weather.

Weather	Sunny	Partly Cloudy	Cloudy	Mostly Cloudy	Raining	Snowing
Sunny	0	0.1	0.3	0.6	1	1
Partly Cloudy	0.1	0	0.1	0.3	0.8	0.8
Cloudy	0.3	0.1	0	0.1	0.7	0.7
Mostly Cloudy	0.6	0.3	0.1	0	0.4	0.4
Raining	1	0.8	0.7	0.4	0	0.6
Snowing	1	0.8	0.7	0.4	0.6	0

Table F.2: Distances for different weather conditions

F.4 User Survey Questionnaire

The questionnaire from the user survey is divided into three sections. In the first two sections the users answer the same questions for the baseline application and the final prototype. Afterwards, the user is asked to answer demographic questions and states, whether she would have chosen the same stereotype as the system proposed.

Likert Scale Statements

Each of the questions (except for question eighteen, which could be answered in a text field) are answered on a 1-7 point Likert (discrete) scale. Instead of the actual values, text representations were shown. This time we used a reversed scale, it is mapped as follows:

1 Strongly agree | 2 Mostly agree | 3 Somewhat agree | 4 Neutral | 5 Somewhat disagree | 6 Mostly disagree | 7 Strongly disagree

- The recommended products fitted my preferences.
- The recommended products were in line with my provided scenario.
- The usage of the application was intuitive and easy to understand.
- I understand how the recommendations were created.
- The system handles my information trustworthy.
- The system helped me make my decision.
- I liked the application's design.
- I would use the application again.
- Do you have any comments you would like to add, regarding this application?

Demographic and Regular Questions

All other questions including demographic questions are presented below. First the user's task is presented in bold letters and then she can choose between the possible answers. The user receives this questionnaire after having completed the survey about the baseline as well as about the final prototype:

Question How old are you?

Answer <20 | 20-25 | 25-30 | 30-35 | 35-40 | 40-45 | 45-50 | 50-55 | 55-60 | 60-65 | 65-70 | 70-75 | 75-80 | 80-85 | 85-90 | >90

Question Which was the number you received from the system?

Answer Text field

Question Are you familiar with the usage of android devices?

Answer Yes | No

Question Do you use a computer or a laptop for online shopping?

Answer Yes | No

Question If "yes" how often?

Answer Several times a week | Once a week | Several times a month | Once a month | Rarely

Question Do you use your smartphone or tablet to shop for clothes?

Answer Yes | No

Question If "yes" how often?

Answer Several times a week | Once a week | Several times a month | Once a month | Rarely

Question Which of the systems do you like more?

Answer System 1 | both equal | System 2

Question Do you have any comments you would like to add?

Answer Text field

Question Which is your gender?

Answer Male | Female

Question Take a look at the following pictures. Which of the fashion styles suits you best?

Answer *The user may select one of the nine pictures of the stereotypes (the male or female versions are displayed depending on the user's gender)*

Bibliography

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.

[Adomavicius and Tuzhilin, 2011] Adomavicius, G. and Tuzhilin, A. (2011). Context-Aware Recommender Systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 217–253. Springer US.

[Alodhaibi et al., 2011] Alodhaibi, K., Brodsky, A., and Mihaila, G. A. (2011). A randomized algorithm for maximizing the diversity of recommendations. In *Proceedings of the Annual Hawaii International Conference on System Sciences*.

[Amatriain, 2013] Amatriain, X. (2013). *Big & personal: data and models behind netflix recommendations*, page 6. ACM.

[Ambrosini et al., 1997] Ambrosini, L., Cirillo, V., and Micarelli, A. (1997). A hybrid architecture for user-adapted information filtering on the World Wide Web. In *User Modeling*, pages 59–61. Springer.

[Anand and Mobasher, 2007] Anand, S. S. and Mobasher, B. (2007). Contextual Recommendation. In *From web to social web: Discovering and deploying user and content profiles*, pages 142–160. Springer Berlin Heidelberg.

[Angerosa, 2011] Angerosa, O. (2011). Fashion Stereotypes. `http://people.rit.edu/ona8039/wf/stage3/index.html`. Last Access: 2013-11-04.

[Ardissono et al., 2001] Ardissono, L., Console, L., and Torre, I. (2001). An adaptive system for the personalized access to news. *AI communications*, 14(3):129–147.

[Ardissono and Goy, 2000] Ardissono, L. and Goy, A. (2000). Tailoring the interaction with users in web stores. *User Modeling and User-Adapted Interaction*, 10(4):251–303.

[Ardissono et al., 1999] Ardissono, L., Goy, A., Meo, R., Petrone, G., Console, L., Lesmo, L., Simone, C., and Torasso, P. (1999). A configurable system for the construction of adaptive virtual stores. *World Wide Web*, 2(3):143–159.

[Argo et al., 2005] Argo, J. J., Dahl, D. W., and Manchanda, R. V. (2005). The influence of a mere social presence in a retail context. *Journal of Consumer Research*, 32(2):207–212.

[Bader et al., 2011] Bader, R., Wörndl, W., Karitnig, A., and Leitner, G. (2011). *Designing an Explanation Interface for Proactive Recommendations in Automotive Scenarios*, page 13. Springer.

[Baltrunas et al., 2011] Baltrunas, L., Ludwig, B., Peer, S., and Ricci, F. (2011). Context-aware places of interest recommendations for mobile users. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, pages 531–540. Springer.

[Baltrunas et al., 2012] Baltrunas, L., Ludwig, B., Peer, S., and Ricci, F. (2012). Context Relevance Assessment and Exploitation in Mobile Recommender Systems. *Personal Ubiquitous Comput.*, 16(5):507–526.

[Baltrunas and Ricci, 2009] Baltrunas, L. and Ricci, F. (2009). Context-based splitting of item ratings in collaborative filtering. In *RecSys'09 - Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 245–248, New York City, NY, USA.

[Bellotti et al., 2008] Bellotti, V., Price, B., Rasmussen, P., Roberts, M., Schiano, D. J., Walendowski, A., Begole, B., Chi, E. H., Ducheneaut, N., Fang, J., Isaacs, E., King, T., Newman, M. W., and Partridge, K. (2008). Activity-Based Serendipitous Recommendations with the Magitti Mobile Leisure Guide. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, pages 1157–1166.

[Bettini et al., 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180.

[Böhmer et al., 2013] Böhmer, M., Ganev, L., and Krüger, A. (2013). AppFunnel: A Framework for Usage-centric Evaluation of Recommender Systems That Suggest Mobile Applications. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 267–276, New York, NY, USA. ACM.

[Bohnert et al., 2008] Bohnert, F., Zukerman, I., Berkovsky, S., Baldwin, T., and Sonenberg, L. (2008). Using collaborative models to adaptively predict visitor locations in museums. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 42–51. Springer.

[Bortz and Doering, 2006] Bortz, J. and Doering, N. (2006). *Forschungsmethoden und Evaluation: f{ü}r Human- und Sozialwissenschaftler*. Springer-Lehrbuch : Bachelor, Master. Springer.

[Bouzeghoub et al., 2009] Bouzeghoub, A., Do, K. N., and Wives, L. (2009). Situation-aware adaptive recommendation to assist mobile users in a campus environment. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 503–509. IEEE.

[Braunhofer et al., 2014] Braunhofer, M., Elahi, M., Ge, M., and Ricci, F. (2014). Context dependent preference acquisition with personality-based active learning in mobile recommender systems. In *Learning and Collaboration Technologies. Technology-Rich Environments for Learning and Collaboration*, pages 105–116. Springer.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc.

[Bridge and Ferguson, 2002] Bridge, D. and Ferguson, A. (2002). Diverse Product Recommendations Using an Expressive Language for Case Retrieval. *Advances in Case-Based Reasoning*, pages 43–75.

[Briguez et al., 2014] Briguez, C., Budán, M., Deagustini, C., Maguitman, A., Capobianco, M., and Simari, G. (2014). *Argument-based mixed recommenders and their application to movie suggestion*, page 16. Elsevier.

[Brusilovsky and Millán, 2007] Brusilovsky, P. and Millán, E. (2007). User Models for Adaptive Hypermedia and Adaptive Educational Systems. In *The adaptive web*, pages 3–53. Springer Verlag.

[Burke, 2002] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

[Burke et al., 1996] Burke, R. D., Hammond, K. J., and Young, B. C. (1996). Knowledge-Based Navigation of Complex Information Spaces. *Proceedings of the national conference on artificial intelligence*, 462:468.

[Burke et al., 1997] Burke, R. D., Hammond, K. J., and Young, B. C. (1997). The FindMe Approach to Assisted Browsing. *IEEE Expert*, 12(4):32–40.

[Butz and Krüger, 2014] Butz, A. and Krüger, A. (2014). *Mensch-Maschine-Interaktion*. De Gruyter Oldenbourg.

[Carenini and Moore, 2006] Carenini, G. and Moore, J. (2006). *Generating and evaluating evaluative arguments*, page 28. Elsevier.

[Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical report, Dartmouth College, Computer Science, Hanover, NH, USA.

[Chen, 2013] Chen, H. (2013). *A Fashion Recommendation System Based on The Wisdom of Crowds*. PhD thesis, Waseda University.

[Chen and Pu, 2005] Chen, L. and Pu, P. (2005). Trust building in recommender agents. In *Proceedings of the Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces at the 2nd International Conference on E-Business and Telecommunication Networks*, pages 135–145. Citeseer.

[Chen and Pu, 2006] Chen, L. and Pu, P. (2006). Evaluating Critiquing-Based Recommender Agents. *Proceedings of the National Conference on Artificial Intelligence*, 21(1):157.

[Chen and Pu, 2007a] Chen, L. and Pu, P. (2007a). Hybrid critiquing-based recommender systems. In *Proceedings of the 12th international conference on Intelligent user interfaces - IUI '07*, pages 22–31. ACM, New York, New York, USA.

[Chen and Pu, 2007b] Chen, L. and Pu, P. (2007b). The Evaluation of a Hybrid Critiquing System with Preference-Based Recommendations Organization. In *Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07*, pages 169–172. ACM, New York, New York, USA.

[Chen and Pu, 2009] Chen, L. and Pu, P. (2009). Interaction Design Guidelines on Critiquing-Based Recommender Systems. *User Modeling and User-Adapted Interaction*, 19(3):167–206.

[Chen and Pu, 2011] Chen, L. and Pu, P. (2011). *Critiquing-based recommenders: survey and emerging trends*, page 26. Springer.

[Chen and Pu, 2012] Chen, L. and Pu, P. (2012). Critiquing-Based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction*, 22(1-2):125–150.

[Chin, 1989] Chin, D. N. (1989). KNOME: Modeling what the user knows in UC. In *User models in dialog systems*, pages 74–107. Springer.

[Ciaramella et al., 2010] Ciaramella, A., Cimino, M., Lazzerini, B., and Marcelloni, F. (2010). Using context history to personalize a resource recommender via a genetic algorithm. In *Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, ISDA'10*, pages 965–970, Cairo, Egypt.

[Codina et al., 2013] Codina, V., Ricci, F., and Ceccaroni, L. (2013). Semantically-enhanced pre-filtering for context-aware recommender systems. In *ACM International Conference Proceeding Series*, pages 15–18, Rome, Italy.

[Cramer et al., 2008] Cramer, H., Evers, V., Ramlal, S., Someren, M., Rutledge, L., Stash, N., Aroyo, L., and Wielinga, B. (2008). The Effects of Transparency on Trust in and Acceptance of a Content-based Art Recommender. *User Modeling and User-Adapted Interaction*, 18(5):455–496.

[Czarkowski, 2006] Czarkowski, M. (2006). A scrutable adaptive hypertext. In *A Scrutable Adaptive Hypertext*. University of Sydney.

[Dao et al., 2012] Dao, T. H., Jeong, S. R., and Ahn, H. (2012). A Novel Recommendation Model of Location-based Advertising: Context-Aware Collaborative Filtering Using GA Approach. *Expert Systems with Applications*, 39(3):3731–3739.

[De Pessemier et al., 2013] De Pessemier, T., Dooms, S., Vanhecke, K., Matté, B., Meyns, E., and Martens, L. (2013). Context-aware Recommendations through Activity Recognition. In Krempels, K.-H. and Stocker, A., editors, *WEBIST*, pages 481–490. SciTePress.

[Dey, 2001] Dey, A. K. (2001). Understanding and Using Context. *Personal and ubiquitous computing*, 5(1):4–7.

[Emrich et al., 2014] Emrich, A., Theobalt, A., Leonhardt, F., Knoch, S., Werth, D., and Loos, P. (2014). A pervasive mobile assistance system for health and fitness scenarios. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 2898–2907. IEEE.

[ExtremeElectronics, 2015] ExtremeElectronics (2015). OpenWeatherMap current weather and forecast. `http://openweathermap.org/`. Last Access: 2015-06-12.

[Facebook, 2015] Facebook (2015). Facebook Graph API. `https://developers.facebook.com/docs/graph-api`. Last Access: 2015-10-25.

[Felfernig and Burke, 2008] Felfernig, A. and Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, page 3. ACM.

[Field and Hole, 2002] Field, A. and Hole, G. (2002). *How to design and report experiments*. Sage.

[Finin, 1989] Finin, T. W. (1989). GUMS—A general user modeling shell. In *User models in dialog systems*, pages 411–430. Springer.

[Fling, 2009] Fling, B. (2009). *Mobile Design and Development: Practical concepts and techniques for creating mobile sites and web apps*. O'Reilly Media, Inc., first edit edition.

[Fogg, 1998] Fogg, B. J. (1998). Persuasive computers: perspectives and research directions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 225–232. ACM Press/Addison-Wesley Publishing Co.

[Froehlich et al., 2006] Froehlich, J., Chen, M. Y., Smith, I. E., and Potter, F. (2006). Voting with your feet: An investigative study of the relationship between place visit behavior and preference. In *UbiComp 2006: Ubiquitous Computing*, pages 333–350. Springer.

[Gauch et al., 2007] Gauch, S., Speretta, M., Chandramouli, A., and Micarelli, A. (2007). User Profiles for Personalized Information Access. In *The adaptive web*, pages 54–89.

[Gavalas and Kenteris, 2011] Gavalas, D. and Kenteris, M. (2011). A web-based pervasive recommendation system for mobile tourist guides. *Personal and Ubiquitous Computing*, 15(7):759–770.

[Gerrig and Zimbardo, 2008] Gerrig, R. J. and Zimbardo, P. (2008). Psychologie (18. Auflage). *Hallbergmoos: Pearson Deutschland GmbH*.

[Goeker and Thompson, 2000] Goeker, M. H. and Thompson, C. A. (2000). Personalized Conversational Case-Based Recommendation. In *Advances in Case-Based Reasoning*, number 2000, pages 99–111. Springer.

[Goodman et al., 2012] Goodman, E., Kuniavsky, M., and Moed, A. (2012). *Observing the User Experience: A Practitioner's Guide to User Research*. Interactive Technologies. Elsevier Science.

[Google, 2013] Google (2013). Google Shopping API. `https://developers.google.com/shopping-search/`. Last Access: 2013-08-12.

[Grabner-Kräuter and Kaluscha, 2003] Grabner-Kräuter, S. and Kaluscha, E. A. (2003). Empirical research in on-line trust: a review and critical assessment. *International Journal of Human-Computer Studies*, 58(6):783–812.

[Gregory et al., 1982] Gregory, W. L., Cialdini, R. B., and Carpenter, K. M. (1982). Self-relevant scenarios as mediators of likelihood estimates and compliance: Does imagining make it so? *Journal of Personality and Social Psychology*, 43(1):89–99.

[Hammer et al.,] Hammer, S., Seiderer, A., André, E., Rist, T., Kastrinaki, S., Hondrou, C., Raouzaiou, A., Karpouzis, K., and Kollias, S. Design of a Lifestyle Recommender system for the Elderly: Requirement Gatherings in Germany and Greece.

[Hassenzahl, 2008] Hassenzahl, M. (2008). User experience (UX): towards an experiential perspective on product quality. In *Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 11–15. ACM.

[Hassenzahl, 2010] Hassenzahl, M. (2010). Experience design: Technology for all the right reasons. *Synthesis Lectures on Human-Centered Informatics*, 3(1):1–95.

[Herlocker, 1999] Herlocker, J. (1999). An empirical study on the persuasiveness of fact-based explanations for recommender systems. Technical report, University of Minnesota.

[Hinze and Buchanan, 2005] Hinze, A. and Buchanan, G. (2005). Context-awareness in mobile tourist information systems: challenges for user interaction.

[Horozov et al., 2006] Horozov, T., Narasimhan, N., and Vasudevan, V. (2006). Using location for personalized POI recommendations in mobile environments. In *Applications and the Internet, 2006. SAINT 2006. International Symposium on*, pages 6—-pp. IEEE.

[Iyengar and Lepper, 2000] Iyengar, S. S. and Lepper, M. R. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology*, 79(6):995–1006.

[Jannach and Kreutler, 2007] Jannach, D. and Kreutler, G. (2007). Rapid development of knowledge-based conversational recommender applications with advisor suite. *Journal of Web Engineering*, 6(2):165.

[Jannach et al., 2010] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edition.

[Jones and Marsden, 2006] Jones, M. and Marsden, G. (2006). *Mobile Interaction Design*. Wiley.

[Jung, 2009] Jung, J. J. (2009). Contextualized Mobile Recommendation Service Based on Interactive Social Network Discovered from Mobile Users. *Expert Systems with Applications*, 36(9):11950–11956.

[Kim et al., 2004] Kim, C. Y., Lee, J. K., and Advanced, K. (2004). V ISCORS : A Visual-Content Recommender for the Mobile Web. *Intelligent systems, IEEE*, 19(6):32–39.

[Knijnenburg et al., 2012] Knijnenburg, B. P., Bostandjiev, S., O'Donovan, J., and Kobsa, A. (2012). Inspectability and Control in Social Recommenders. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 43–50, New York, NY, USA. ACM.

[Kofod-Petersen and Aamodt, 2003] Kofod-Petersen, A. and Aamodt, A. (2003). Case-based situation assessment in a mobile context-aware system. In *Artificial Intelligence in Mobile Systems*, AIMS 2003, Seattle, WA, USA.

[Konstan and Riedl, 2012] Konstan, J. A. and Riedl, J. (2012). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123.

[Krüger et al., 2007] Krüger, A., Baus, J., Heckmann, D., Kruppa, M., and Wasinger, R. (2007). Adaptive mobile guides. In *The adaptive web*, pages 521–549. Springer.

[Lamche, 2014] Lamche, B. (2014). Improving Mobile Recommendations through Context-Aware User Interaction. In *Proc. UMAP Doctoral Consortium, 22nd Conference on User Modeling, Adaption and Personalization*, volume 22, pages 1–6, Aalborg, Denmark.

[Lamche et al., 2014a] Lamche, B., Adigüzel, U., and Wörndl, W. (2014a). Interactive Explanations in Mobile Shopping Recommender Systems. In *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, page 14.

[Lamche et al., 2014b] Lamche, B., Pollok, E., Wörndl, W., and Groh, G. (2014b). Evaluating the Effectiveness of Stereotype User Models for Recommendations on Mobile Devices. In *Proceedings of the Joint Workshop on Personalized Information Access (PIA 2014), in conjunction with the 22nd conference on User Modeling, Adaptation and Personalization (UMAP 2014)*, pages 1—–6, Aalborg, Denmark.

[Lamche et al., 2015a] Lamche, B., Rödl, Y., Hauptmann, C., and Wörndl, W. (2015a). Context-Aware Recommendations for Mobile Shopping. *ACM RecSys Workshop on Location-Aware Recommendations*.

[Lamche et al., 2015b] Lamche, B., Sahinagic, N., and Wörndl, W. (2015b). User Interaction Design for Mobile Product Recommender Systems. In *Proceedings of the 11th International Conference on Web Information Systems and Technology (WEBIST 2015)*, pages 1–11, Lisbon.

[Lamche et al., 2014c] Lamche, B., Trottmann, U., and Wörndl, W. (2014c). Active Learning Strategies for Exploratory Mobile Recommender Systems. In *CaRR workshop, ECIR*, Amsterdam.

[Lazar et al., 2010] Lazar, J., Feng, J. H., and Hochheiser, H. (2010). *Research methods in human-computer interaction*. John Wiley & Sons.

[Lee and Kwon, 2013] Lee, H. and Kwon, J. (2013). Situation and social awareness-based personalized recommendation service in pervasive computing environment. In *IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2013*, pages 682–687.

[Lee and Lee, 2007] Lee, J. S. and Lee, J. C. (2007). Context awareness by case-based reasoning in a music recommendation system. *Lecture Notes in Computer Science*, 4836:45–58.

[Liang et al., 2006] Liang, T.-P., Lai, H.-J., and Ku, Y.-C. (2006). Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems*, 23(3):45–70.

[Lunden, 2013] Lunden, I. (2013). Apple's iBeacon Comes To Retailers Via Shopkick's ShopBeacon.

[Luo, 2005] Luo, X. (2005). How Does Shopping With Others Influence Impulsive Purchasing? *Journal of Consumer Psychology*, 15(4):288–294.

[Marchionini, 2006] Marchionini, G. (2006). Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4):41–46.

[McGinty and Reilly, 2011] McGinty, L. and Reilly, J. (2011). On the Evolution of Critiquing Recommenders. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 419–453. Springer US.

[McGinty and Smyth, 2003a] McGinty, L. and Smyth, B. (2003a). On the Role of Diversity in Conversational Recommender Systems. In *Case-based reasoning research and development*, pages 276–290. Springer Berlin Heidelberg.

[McGinty and Smyth, 2003b] McGinty, L. and Smyth, B. (2003b). Tweaking Critiquing. *Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 20–27.

[Mehra et al., 2006] Mehra, S., Werkhoven, P., and Worring, M. (2006). Navigating on handheld displays: Dynamic versus static peephole navigation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(4):448–457.

[Micarelli and Sciarrone, 2004] Micarelli, A. and Sciarrone, F. (2004). Anatomy and empirical evaluation of an adaptive web-based information filtering system. *User Modeling and User-Adapted Interaction*, 14(2-3):159–200.

[Miller et al., 2003] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. (2003). MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 263–266, New York, NY, USA. ACM.

[Miller et al., 2004] Miller, B. N., Konstan, J. A., and Riedl, J. (2004). PocketLens: Toward a personal recommender system. *ACM Transactions on Information Systems (TOIS)*, 22(3):437–476.

[Na and Agnhage, 2013] Na, Y. and Agnhage, T. (2013). Relationship between the preference styles of music and fashion and the similarity of their sensibility. *International Journal of Clothing Science and Technology*, 25(2):109–118.

[Neidhardt et al., 2014] Neidhardt, J., Schuster, R., Seyfang, L., and Werthner, H. (2014). Eliciting the users' unknown preferences. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 309–312. ACM.

[Nguyen et al., 2004] Nguyen, Q. N., Ricci, F., and Cavada, D. (2004). Critique-based Recommendations for Mobile Users : GUI Design and Evaluation. *Third Workshop on "HCI in Mobile Guides" in Conjunction with Sixth International Conference on Human Computer Interaction with Mobile Devices and Services.*

[Panniello and Gorgoglione, 2011] Panniello, U. and Gorgoglione, M. (2011). A contextual modeling approach to context-aware recommender systems. In *CEUR Workshop Proceedings*, volume 791, Chicago, IL, USA.

[Panniello et al., 2014] Panniello, U., Tuzhilin, A., and Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65.

[Park et al., 2007] Park, M.-H., Hong, J.-H., and Cho, S.-B. (2007). Location-based recommendation system using bayesian user's preference model in mobile devices. In *Ubiquitous Intelligence and Computing*, pages 1130–1139. Springer.

[Perugini et al., 2004] Perugini, S., Gonçalves, M. A., and Fox, E. A. (2004). Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2):107–143.

[Polatidis and Georgiadis, 2013] Polatidis, N. and Georgiadis, C. K. (2013). Mobile recommender systems: An overview of technologies and challenges. In *Informatics and Applications (ICIA), 2013 Second International Conference on*, pages 282–287. IEEE.

[Pu et al., 2011a] Pu, P., Chen, L., and Hu, R. (2011a). A User-centric Evaluation Framework for Recommender Systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 157–164, New York, NY, USA. ACM.

[Pu et al., 2012] Pu, P., Chen, L., and Hu, R. (2012). Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4-5):317–355.

[Pu et al., 2011b] Pu, P., Faltings, B., Chen, L., Zhang, J., and Viappiani, P. (2011b). Usability Guidelines for Product Recommenders Based on Example Critiquing Research. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 511–545. Springer, Boston, MA.

[Pushpa and Venkataram, 2011] Pushpa, P. V. and Venkataram, P. (2011). Context aware M-commerce services: C-IOB model approach. In *8th International Conference on Information, Communications and Signal Processing*, ICICS, Singapore.

[Reutskaja and Hogarth, 2009] Reutskaja, E. and Hogarth, R. M. (2009). Satisfaction in choice as a function of the number of alternatives: When "goods satiate". *Psychology and Marketing*, 26(3):197–203.

[Ricci, 2010] Ricci, F. (2010). Mobile Recommender Systems. *Information Technology & Tourism*, 12(3):205–231.

[Ricci, 2013] Ricci, F. (2013). Contextualizing Useful Recommendations. `http://www.inf.unibz.it/~ricci/Slides/Context-UMAP-2012-Ricci.pdf`. Last Access: 2013-12-15.

[Ricci et al., 2002] Ricci, F., Arslan, B., Mirzadeh, N., and Venturini, A. (2002). ITR: a case-based travel advisory system. In *Advances in Case-Based Reasoning*, pages 613–627. Springer.

[Ricci and Nguyen, 2007] Ricci, F. and Nguyen, Q. N. (2007). Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System. *Intelligent Systems*, 22(3):22–29.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to Recommender Systems Handbook. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 1–35. Springer.

[Ricci et al., 2005] Ricci, F., Woeber, K., and Zins, A. (2005). Recommendations by collaborative browsing. *Information and Communication Technologies in Tourism 2005*, pages 172–182.

[Rich, 1979a] Rich, E. (1979a). Building and exploiting user models. In *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 2*, pages 720–722. Morgan Kaufmann Publishers Inc.

[Rich, 1979b] Rich, E. (1979b). User Modeling via Stereotypes. *Cognitive science*, 354(3597):329–354.

[Rich, 1983] Rich, E. (1983). Users are individuals: individualizing user models. *International journal of man-machine studies*, 18(3):199–214.

[Rogers et al., 2011] Rogers, Y., Sharp, H., and Preece, J. (2011). *Interaction Design: Beyond Human - Computer Interaction*. Interaction Design: Beyond Human-computer Interaction. Wiley.

[Rubens et al., 2011] Rubens, N., Kaplan, D., and Sugiyama, M. (2011). Active Learning in Recommender Systems. In *Recommender Systems Handbook*, pages 735–767. Springer, Boston, MA.

[Ruotsalo et al., 2013] Ruotsalo, T., Haav, K., Stoyanov, A., Roche, S., Fani, E., Deliai, R., Mäkelä, E., Kauppinen, T., and Hyvönen, E. (2013). SMARTMUSEUM: A mobile recommender system for the Web of Data. *Web semantics: Science, services and agents on the world wide web*, 20:50–67.

[Savage et al., 2012] Savage, N. S., Baranski, M., Chavez, N. E., and Höllerer, T. (2012). *I'm feeling loco: A location based context aware recommendation system*. Springer.

[Shani and Gunawardana, 2011] Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Shearin and Lieberman, 2001] Shearin, S. and Lieberman, H. (2001). Intelligent profiling by example. *Proceedings of the 6th international conference on Intelligent user interfaces - IUI '01*, pages 145–151.

[Shimazu, 2002] Shimazu, H. (2002). ExpertClerk : A Conversational Case-Based Reasoning Tool for Developing Salesclerk Agents in E-Commerce Webshops. *Artificial Intelligence Review*, 18(3-4):223–244.

[Shneiderman, 2015] Shneiderman, B. (2015). The New Computing. `http://ubiquity.acm.org/article.cfm?id=763933`. Last Access: 2015-02-07.

[Shneiderman and Plaisant, 2005] Shneiderman, S. B. and Plaisant, C. (2005). Designing the user interface 4 th edition. *ed: Pearson Addison Wesley, USA*.

[Sinha and Swearingen, 2002] Sinha, R. and Swearingen, K. (2002). The Role of Transparency in Recommender Systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 830–831, New York, NY, USA. ACM.

[Sleeman, 1985] Sleeman, D. (1985). UMFE: a user modelling front-end subsystem. *International Journal of Man-Machine Studies*, 23(1):71–88.

[Smyth, 2007] Smyth, B. (2007). Case-Based Recommendation. In *The adaptive web*, volume 4321, chapter The adapti, pages 342–376. Springer Berlin Heidelberg.

[Smyth and McClave, 2001] Smyth, B. and McClave, P. (2001). Similarity vs. Diversity. In *Case-Based Reasoning Research and Development*, pages 347–361. Springer Berlin Heidelberg.

[Sø rmo et al., 2005] Sø rmo, F., Cassens, J., and Aamodt, A. (2005). Explanation in Case-Based Reasoning—Perspectives and Goals. *Artif. Intell. Rev.*, 24(2):109–143.

[Statista, 2015] Statista (2015). Number of apps available in leading app stores as of July 2014. `http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`. Last Access: 2015-07-29.

[Stewart et al., 2008] Stewart, R., Scott, G., and Zelevinsky, V. (2008). Idea navigation: structured browsing for unstructured text. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1789–1792. ACM.

[Swearingen and Sinha, 2001] Swearingen, K. and Sinha, R. (2001). Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems*, volume 13, pages 1–11. Citeseer.

[Tidwell, 2010] Tidwell, J. (2010). *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media.

[Tintarev and Masthoff, 2007a] Tintarev, N. and Masthoff, J. (2007a). A Survey of Explanations in Recommender Systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 801–810, Washington, DC, USA. IEEE Computer Society.

[Tintarev and Masthoff, 2007b] Tintarev, N. and Masthoff, J. (2007b). Effective Explanations of Recommendations: User-centered Design. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, pages 153–156, New York, NY, USA. ACM.

[Tintarev and Masthoff, 2011] Tintarev, N. and Masthoff, J. (2011). *Designing and Evaluating Explanations for Recommender Systems*, page 33. Springer.

[Tintarev and Masthoff, 2012] Tintarev, N. and Masthoff, J. (2012). Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):399–439.

[Van Setten et al., 2004] Van Setten, M., Pokraev, S., and Koolwaaij, J. (2004). Context-aware recommendations in the mobile tourist application COMPASS. In *Adaptive hypermedia and adaptive web-based systems*, pages 235–244. Springer.

[Viappiani et al., 2006] Viappiani, P., Faltings, B., and Pu, P. (2006). Evaluating preference-based search tools: a tale of two approaches. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 21, page 205. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Vig et al., 2009] Vig, J., Sen, S., and Riedl, J. (2009). *Tagsplanations: explaining recommendations using tags*, page 10. ACM.

[Virvou and Tsiriga, 2003] Virvou, M. and Tsiriga, V. (2003). Adaptive Tutoring Based on the Student Model of a Webbased ICALL. *Sept. 25th.*

[von Alan et al., 2004] von Alan, R. H., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

[Wang et al., 2011] Wang, L. C., Meng, X. W., and Zhang, Y. J. (2011). A heuristic approach to social network-based and context-aware mobile services recommendation. *Journal of Convergence Information Technology*, 6(10):339–346.

[Wasinger et al., 2013] Wasinger, R., Wallbank, J., Pizzato, L., Kay, J., Kummerfeld, B., Böhmer, M., and Krüger, A. (2013). Scrutable User Models and Personalised Item Recommendation in Mobile Lifestyle Applications. In *User Modeling, Adaptation, and Personalization*, volume 7899 of *Lecture Notes in Computer Science*, pages 77–88. Springer Berlin Heidelberg.

[Woerndl and Groh, 2007] Woerndl, W. and Groh, G. (2007). Utilizing physical and social context to improve recommender systems. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops*, WI-IAT Workshops, pages 123–128, Silicon Valley, CA, USA.

[Woerndl and Lamche, 2015] Woerndl, W. and Lamche, B. (2015). User Interaction with Context-aware Recommender Systems on Smartphones. *icom*, 14(1):19–28.

[Woerndl et al., 2012] Woerndl, W., Lerchenmueller, B., and Schulze, F. (2012). Influencing Factors for User Context in Proactive Mobile Recommenders. *ABIS 2012.*

[Woerndl et al., 2007] Woerndl, W., Schueller, C., and Wojtech, R. (2007). A hybrid recommender system for context-aware recommendations of mobile applications. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 871–878. IEEE.

[Yahoo, 2015] Yahoo (2015). Weather - Yahoo Developer Network. `https://developer.yahoo.com/weather/`. Last Access: 2015-06-10.

[Yap et al., 2007] Yap, G.-E., Tan, A.-H., and Pang, H.-H. (2007). Discovering and exploiting causal dependencies for robust mobile context-aware recommenders. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):977–992.

[Zagel, 2015] Zagel, C. T. (2015). *Gaining Competitive Advantage Through Experiential Self-Service Systems in Retail Environments.* PhD thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg.

[Zalando, 2015] Zalando (2015). Zalando API. `https://api.zalando.com/`. Last Access: 2015-05-30.

[Zhang et al., 2008] Zhang, J., Jones, N., and Pu, P. (2008). A Visual Interface for Critiquing-Based Recommender Systems. *Proceedings of the 9th ACM conference on Electronic commerce - EC '08*, pages 230–239.

[Zhang et al., 2013] Zhang, Z., Shang, S., Kulkarni, S. R., and Hui, P. (2013). Improving augmented reality using recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 173–176. ACM.

List of Figures

List of Tables