FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Implementation and Evaluation of a Context-Aware Mobile Shopping Recommender System

Yurong Tao

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Implementation and Evaluation of a Context-Aware Mobile Shopping Recommender System

TODO: Titel der Abschlussarbeit

Author:	Yurong Tao
Supervisor:	Prof. Dr. Johann Schlichter
Advisor:	Béatrice Lamche, M.Sc.
Submission Date:	August 01, 2014

I assure the single handed composition of this master's thesis in informatik only supported by declared resources.

Munich, August 01, 2014 Yurong Tao

Acknowledgments

Abstract

Contents

Acknowledgments **iv**

Abstract **v**

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Goals . . . 2
- 1.3 Outline . . . 3

2 Foundations **4**
- 2.1 Recommender System . . . 4
  - 2.1.1 Case-Based Recommender System . . . 5
  - 2.1.2 Active Learning Recommender System . . . 8
  - 2.1.3 Conversation-based Active Learning . . . 9
  - 2.1.4 Context-Aware Recommender Systems . . . 10
  - 2.1.5 Mobile Recommender System . . . 16
  - 2.1.6 Baseline System . . . 18
  - 2.1.7 Methodology for Developing CARS . . . 20

3 Build the System **21**
- 3.1 Acquiring Context Relevance . . . 21
  - 3.1.1 Analysis of Context Relevance . . . 22
- 3.2 The Proposed Approach . . . 23
- 3.3 Towards Case-Based Recommender Systems . . . 24
- 3.4 Case Model . . . 25
- 3.5 Similarity Assessment . . . 26
- 3.6 Explanation Generation . . . 27
- 3.7 Interaction and Interface Design . . . 28

4 Evaluation **31**
- 4.1 Motivation and Goals . . . 31

4.2 Data Set 32
4.2.1 Data Set for Clothes Items 32
4.2.2 Data Set for Context Case Base 33
4.3 Test Setup 34
4.3.1 Testing Framework 34
4.3.2 Testing Procedure 35
4.4 Results 37
4.4.1 Participants 38
4.4.2 Overview 38
4.4.3 Prediction Accuracy 39
4.4.4 Decision Efficiency 39
4.4.5 Context Benefits 42
4.4.6 Explanation Benefits 43
4.4.7 System Preference 45
4.4.8 Informal Feedback 46
4.4.9 Correlation Analysis 47
4.5 Discussion 48

5 Conclusions and Future Work **50**

1 Introduction

In this Master's Thesis, a context-aware mobile shopping recommender system is implemented and evaluated. A methodology for building context-aware recommender systems was adopted and adapted to the system. Following the methodology, a recommendation approach and the reasoning behind why this specific approach was chosen will be given. The development and design process together with the evaluation process will be presented.

This chapter will shed light on the motivations behind the developed system and the goals set out to meet. The last section will give a brief outline on how this thesis is structured.

1.1 Motivation

In this thesis a mobile shopping recommender system developed for recent touch-based Android phone handsets will be displayed. In this system, contextual information was integrated into the recommendation process so as to provide context-aware recommendations.

Mobile platform was selected in this thesis because it is the trend. Mobile computing has caught the attention of the research community for quite some time. Many experimental systems and applications have been developed but were not vastly put in real use because of the limitation of the mobile devices and the infrastructure outside. However, around six years ago, the introduction of new mobile platforms such as iPhone and Android changed everything drastically. The new touch-based interaction method and the improvement of the computing power of mobile devices bring new possibilities to application development. The constant development of wireless network bandwidth and the decrease in the price of both the mobile devices and the network fee help to create a large customer base for developers. The reduce in the size of mobile devices also makes it easier to carry them around. According to reports from Market Research firm Gartner, by the year 2016, an estimated 310B downloads and $74B in revenue is predicted from app stores [12].

In particular, Android was selected at the target mobile platform in this thesis. Android is the most popular mobile platform and is still growing very fast - every day more than one million new Android devices are activated worldwide [2]. The openness and the powerful development framework make Android applications be deployable across a wide range of devices. On the other hand, from the developer's perspective, the Android development tools offer a full Java IDE with advanced features for developing, debugging, and packaging Android apps that can efficiently facilitate the development.

In addition, mobile devices are generally becoming an essential part of people′ s daily life. People can carry them around and have access to internet anywhere at anytime. Mobile devices have simplified and changed the way people do business, do shopping, travel and communicate. This feature of mobile also draws the attention to context-aware systems. Especially for recommender systems, which and to what degree can context factors affect people′s perception of the recommended items and how can contextual information be effectively integrated into existing recommender systems need to be further studied.

In mobile exploratory scenarios, the user does not know exactly what she/he is looking for, or she/he might have a general idea of the product to buy (e.g., buy clothes for sports purpose). In the ideal case, the system should be designed not to require any query input from the user at the start of the recommendation session. Instead, to best predict user′s preference, a diverse set of items will be presented so as to ensure that the user can start general and determine the direction to go. However, if the diversity of items increases, further personalization is required for initial recommendation to ensure the accuracy and efficiency of the system. At this time, context-aware information such as weather, budget and shopping intent can be important clues to predicting user′s current interest.

As a result, the focus of this thesis shifted to implementing and evaluating context-aware recommender systems using Active Learning strategies on mobile platforms. To be more specific, a clothes shopping scenario was used in this thesis because it was not much studied but is closely related to people′s daily life.

1.2 Goals

Few studies have been done on the integration of context-aware information into an active learning mobile recommender system [3, 11, 9]. The goal of this paper is to explore if the integration of context-aware information using case-based recommendation approach can improve the user experience of mobile recommender system that uses a conversation-based Active Learning strategy.

The system designed should be flexible enough to include different types of mobile contextual information. In this thesis, four types of context should be considered: physical context, social context, interaction media context and modal context.

The interaction design of the system will follow the Android Design Principles [1] to ensure an integrated experience and user's acceptance of the system. In mobile shopping scenario, users are believe to be less patient, mostly on the move and likely to be distracted easily, contextual information retrieval, thus, should be kept as simple as possible and not too much user input should be required. If possible, automatic detection can be used to minimize user input.

To successfully design and develop the context-aware recommender system, a methodology for developing context-aware recommender system [3] should be used and adapted to the current system. Firstly, context factors relevance will be assessed using a web tool developed for that. Then a case-based recommendation algorithm will be designed and developed based on that. After that, a context-aware mobile recommender system that utilizes the developed algorithm will be developed and evaluated.

1.3 Outline

The thesis is divided into five chapters. The current one (chapter 1), introduces the ideas, motivations and goals behind this thesis.

The second chapter lays a foundation for the system developed in this thesis. It gives a general introduction to recommender system, case-based recommender system, active learning recommender system and context-aware recommender system. The baseline system used for evaluation in this thesis is also introduced in this chapter. Finally, a methodology adopted in this thesis for developing context-aware recommender system is introduced.

The third chapter follows the methodology in chapter two and explains step by step how the system was built. First, an experiment for acquiring context relevance is explained. Then a proposed approach together with the algorithm for integrating contextual information into existing recommender system using Active Learning strategies is explained. Finally, the developed context-aware mobile recommender system prototype Shopper is introduced.

Shopper is evaluated in chapter four. It shows that Shopper received a better evaluation in prediction accuracy, decision efficiency and general satisfaction compared with the similar, but not context-aware baseline system introduced in chapter three.

The thesis ends with a summary of the achievements and discussion of possible directions for future work.

2 Foundations

Based on the motivations and goals discussed in the previous chapter, this chapter will start with a detail discussion of the recommender system researches that informed the design, implementation and evaluation of the developed system.

2.1 Recommender System

With the development of information technology, people can now have access to various kinds of product and services all over the world using internet without any difficulty compared to limited choices in the local shops in the old days. At the same time, people are overwhelmed by the number of options to consider. Thus Recommender Systems (RSs) as a decision support tool has become an important topic in the research study.

Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user [10]. RSs try to provide personalized recommendations based on user preferences and constraints which can be either explicitly expressed as, e.g., ratings of products, or implicitly inferred from, e.g., user actions or context around. For instance, user′s navigation to a particular product page can be interpreted as user′s interest in that product. With the introduction of the concept of context-aware recommender system, context such as location, weather, company can also be used for recommendation prediction.

Different recommendation techniques are used for recommendation and they are usually categorized into three categories [7, 10]:

- **Collaborative Filtering**: RSs using collaborative filtering approach recommend to the active user items that are liked by other users with similar tastes in the past.
- **Content-based**: RSs using content-based approach recommend items that are similar to the ones the current user liked before.
- **Knowledge-based**: RSs using knowledge-based approach exploit knowledge about the customers and the application domain to reason about the items that might be of interest to the user.

In collaborative RSs, recommendation depends on the information of current user and a large amount user/system interaction data is needed to find similar users and thus relevant items to recommend. In content-based RSs, only the current user information is exploited for recommendation and since content-based RSs are usually implemented as classifier systems based on machine learning research, a large enough number of user ratings is needed to train the prediction model. Both techniques are faced with a cold-start problem: not until enough examples (product ratings or pattern of user preferences) are known, the recommendations can not be useful for the user [7, 4].

This problem is even worse for clothing recommender systems. If we look into Amazon website, we can find that there are far fewer ratings for clothes than for other products like cameras or mobile phones. It is because clothing product is featured with fast updating and diverse styles. It is unrealistic to have enough ratings for all the clothing for recommendations in a short period of time. As for the application in this thesis, where a clothing recommendation system for offline shopping scenario was built. Different from online shopping, offline shopping is limited with location and space, which will drastically cut down the number of examples. On the other hand, user′s clothing shopping preferences change a lot with the current contextual conditions like weather, temperature, mood or budget and it is difficult for content-based and collaborative RSs to adapt quickly to those changes. All these considerations motivated the adoption of the third approach, knowledge-based approach. Knowledge-based RSs help to solve all these challenges by exploiting existing knowledge about the specific product domain and explicit user requirements. There is no cold-start problem because all requirements are directly elicited within a recommendation session.

There are two types of knowledge-based RSs: case-based recommender and constraint-based recommender [5]. Case-based recommenders determine recommendations based on past similar cases or in other words, successful solutions, while constraint-based recommenders determine recommendations by exploiting predefined knowledge bases that contain explicit rules about relationship between customer requirements and item features.

2.1.1 Case-Based Recommender System

Case-based reasoning (CBR) is a problem solving methodology that tries to solve new problems by re-using or adapting past solutions stored in past similar cases [6]. A case models a past experience, storing both problem description and solution applied in that context. A case can be as simple as a product. It can also include a product and the context in which the product is bought. All the cases are stored in a case base. In a typical CBR process, when the system is given a new problem to solve, it first searches in the case base for previous similar cases, then reuses or adapts the solutions in those cases to the current problem. After that, the solution together with the problem are retained as a new case in the system case base for the use of future recommendation.

In paper [7], a methodology was introduced to help with applying CBR steps to the recommender systems, in which a framework was created for building CBR RSs. In this framework, a typical case-based recommendation process is composed of six steps: retrieval, reuse, revise, review, retain and iterate, which is similar to the CBR process. Each of these steps will be explained separately below so as to provide a better understanding of this framework.

Retrieval: Retrieval phase, as the first step of the recommendation cycle, is also the main phase and the majority of CBR recommender system. Cases with similar problem descriptions are retrieved in this step using different retrieving strategies.

Reuse: The cases retrieved in the last stage are evaluated and reused in the current problem in this step. There are various ways to reuse the cases. In the simplest cases, the solutions in the retrieved cases will be directly applied to the current problem. In more advanced cases, the retrieved cases can be used as reference set to rank candidate items [9].

Revise: In the revise stage, the reused cases are adapted to better fit to the current problem. For example, if the recommended travel location in the previous case is already closed or is too far away for the current user, then the system can replace it with another similar place that is still open or is nearby.

Review: In the review stage, user can customize or critique the recommended items. For example, the user can tell the system what she/he likes and doesn′t like about a recommended clothes item.

Iterate: Usually in a conversational recommender system, the critiques and customization in the review stage will be given as a feedback to the system so that the system can iteratively update its recommendations to better fit the users′ preferences.

Retain: Finally, at the end of a recommendation session, e.g., an item is bought or an item′s page is viewed, a new case will be created and retained in the case base for future use.

Case Modeling

A case base CB can be decomposed into four sub-components:

$$CB \subseteq X \times U \times S \times E$$

where X is the product/content model, U is the user model, S is the session model and E is the evaluation model. Each case $c = (x, u, s, e) \in CB$ in the case base is consist of four sub-elements x, u, s, e which are instances of the spaces X, U, S, E respectively. However it′s not a must that every case base contains all four components.

Content model (X): The content model describes the product recommended or to be recommended and is usually represented as feature vectors. For example, a clothes item to be recommended in the case base can be represented as an n-dimensional vector space $X = \prod_{i=1}^{n} X_i$. Each X_i represents the set of possible values for a product attribute [8]. An attribute could be the color or price or clothes type.

User model (U): The user model contains the user information, such as, name, address, age or user′s past interactions with the system, such as, the products bought before or the products liked before.

Session model (S): The session model usually contains the information about the recommendation session. It can be the contextual information, such as, weather, temperature, budget, shopping intent, etc.

Evaluation model (E): The evaluation model describes the outcome of the recommendation (e.g., user′s rate).

Similarity Assessment

Retrieval is the majority of case-based recommendation in which cases similar to the current problem will be retrieved. Thus the definition of similarity and the approaches to similarity assessment becomes an important topic and affects the retrieval strategy. For example, if the user submits a query to find similar clothes to the current selected one, the products′ features in each case will be compared to the selected product to determine the similarity. However, if the user submits a query to find clothes to wear in the current season (e.g., summer), then the season information contained in session model in each case will be compared to the current season to determine the similarity.

Similarity assessment between cases or between user query and a candidate case is usually calculated as a weighted sum of similarity between individual feature pairs as is shown in Equation 2.1:

$$Similarity(t, c) = \frac{\sum_{i=1} w_i * sim_i(t_i, c_i)}{\sum_{i=1...n} w_i} \tag{2.1}$$

where the similarity between target query t and candidate case c is calculated as weighted sum of the individual similarities between the corresponding features of t and c, namely t_i and c_i. A unique weight is defined for each feature according to it′s level of importance in the current problem and a unique similarity function $sim_i(t_i, c_i)$ is defined for each feature according to the property of the feature. For example, considering a numeric feature such as clothes size, the maximum similarity is achieved when the size in the target query matches the one in the candidate case. The desirability of the size will decrease equally if it goes either larger or smaller. In this case, the size feature can be measured by symmetric similarity metric like the one in Equation 2.2 [**?**]:

$$sim_{size}(s_t, s_c) = 1 - \frac{|s_t - s_c|}{range(s)} \tag{2.2}$$

On the other hand, considering a numeric feature such as price, if the user queries for clothes of price €50, it means that the user can also afford clothes with price lower than €50, but it may not be the case for clothes with price higher than €50. Thus the price feature can be measured by asymmetric similarity metric like the one in Equation 2.3 [**?**]:

$$sim_{price}(p_t, p_c) = 1 - \frac{|p_t - p_c|}{max(p_t, p_c)} \tag{2.3}$$

To evaluate the similarity of non-numeric features usually requires additional domain knowledge and hard coding the knowledge into the system using a proper a structure, e.g., an ontology.

2.1.2 Active Learning Recommender System

Traditionally, Recommender Systems present items to users because those items are thought to be of interests of the users. However, RSs can also intentionally influence the presentation of items to users so that more user preferences can be elicited and learned. Active Learning (AL) helps to fulfill this goal through actively influencing which items the user is exposed to and learning from user?s feedback [29]. For example, a movie recommender system can first let the user select the genre the user is interested in and then recommend more movies from the genres selected by the user or it can first display a diverse set of movies from different genres and let the user rate or critique on them to get the user?s preferences.

Usually three situations can occur when the user searches for a product: the user knows exactly what to buy; the user has a desire (e.g., want to buy clothes for work) but does not have a clear objective in mind; the user does not know precisely what to buy or just wants [22]. For users in the second and third situation, RSs augmented with AL help the user become more self-aware of their own interests and at the same time collects more information about the user that can be used for better recommendation.

AL is the process of actively selecting training points so as to observe the most informative output. AL methods can be categorized based on their primary goals:

- Uncertainty-based Active Learning: This approach selects training points so as to reduce uncertainty in, for example, output values, the model?s parameters or a decision boundary, etc.

- Error-based Active Learning: This approach selects training points so as to reduce the predictive error.

- Ensemble-based Active Learning: There are usually advantages and disadvantages for each single model. Thus a combination of different models can give a better result through compensating the disadvantages of each other.

2.1.3 Conversation-based Active Learning

Different from standard AL methods in which the goal is to obtain ratings of representative items from users so as to improve the prediction accuracy of underlying model for the entire data set. Conversation-based AL first starts with a general set of items and then iteratively narrow down the scope of candidate items through a series of interaction cycles in which user?s preferences are elicited and learned in various ways until a final desired item is found [29]. A typical example is to first present the user with a diverse set of items and iteratively update and adapt the recommendations based on user?s critiques on item features [30]. In each iteration, the scope of candidate items is narrowed down, because user?s queries are becoming more and more concrete. This process will keep going until the scope is narrowed down to a single item or the user finds the desired item.

This cycle-based conversational approach usually works because it mimics the behavior of offline shopping guide. People are usually unaware of their own interest and don?t know what is out there in the market. Through a conversational interaction, people can first explore the available products and then refine and form their preferences over each cycle. This process is also self-explained enough because user can see how the final recommendation is reached. Such conversational systems include AL by design because user preferences are learned through active interaction. There are three main approaches to perform conversation with the user [29]:

- Case-based Critique: This approach searches for cases similar to user?s query and then update the recommendations based on user?s critiques. During the initialization of the recommendation, no user preference initialization is required. In each iteration, a list of cases will be retrieved and be ranked according to the similarity to current user?s query. The user can then critique on the attributes of the recommended items to further filter out irrelevant items or narrow down the scope of candidate items. There can be positive or negative critiques. Positive critique can be something like "I like color red.", while negative critique can be something like "I don't like jacket.".

- Diversity-based: Although cases are retrieved based on their similarity to the current user?s query, diversity is also an important consideration. In each cycle of the recommendation, the system should present the most representative items in the current search space. If the recommended items are too similar to each other, it might be the case that the user dislikes all of them and have to use more steps to find the items they like. Thus the system should also consider diversity in each iteration and let the user choose the direction to go in the next step. It is especially important when user?s preferences are still not clear.

- Query Editing-based: This approach allows user to repeatedly edit and submit the query in order to get better recommendations. In order to make this process be more efficient, usually, the system can make predictions of the next query or make editing suggestions to the current query so as to save user?s exploration time and help the user to narrow down to their interest area sooner.

2.1.4 Context-Aware Recommender Systems

In traditional cases where people talk to each other, context helps with increasing conversational bandwidth [1]. The current time, the location where the two people meet each other, the gestures, or the companion are all influential factors of the conversation. However, this important ability of context is largely neglected when it comes to human-computer interaction.

With the development of portable computers and wireless communications, people can now carry their personal devices and have access to the mobile network anywhere at anytime. Instead of being used in a steady environment with a predictable environment, the software systems now will be installed on different devices and be used in different environments. With traditional human-computer interaction paradigms, it becomes more difficult for human and computer systems to talk to each other. Also since the customers now can be accessed at anytime through either apps on the mobile phone or other mobile devices like wristband and intelligent glasses, companies now must deliver not just competitive products but also unique, real-time customer experience [32].

Among the researches that have been done in recommender systems, most existing approaches recommend most relevant items to users without taking into account the current context of the user, such as time, location or purpose. However, for recommender systems and especially those on mobile devices, it might not be sufficient to just consider item and user when making recommendations for items in specific fields like vacation package, retailing or movies, because users? preferences may often change according to the current context. For example, users may prefer to visit indoor sight spot when the weather is raining and may prefer to visit museums when they are with their young children. Similarly, in the case of clothes shopping, a user may want to buy coat when the weather is cold even though the user generally like skirt the best. A user may even don?t know what kind of clothes to buy for a certain context (e.g., what clothes to buy for hiking and what clothes to buy for running) and needs the recommender system to assist them in making purchasing decisions. Thus, it is important to incorporate the contextual information into the recommendation process so as to provide better recommendations in different circumstances.

Many researches have already laid the foundation and shown the importance of context in recommender systems. Findings in behavior research shows that decision making is contingent on the context of decision making [2]. Therefore, to increase user satisfaction and prediction accuracy, RSs need to capture the real-time user preferences by integrating context into the recommender systems. Also, a number of ubiquitous computing researchers also share the hypothesis that enabling devices and applications to automatically adapt to changes in context will help to improve the user experience [3]. For example, paper [5] assess the influence of context factors to user ratings and integrate the contextual information in a mobile travel planning tool. It shows that the context-aware system is preferred to a similar variant that did not exploit contextual information in terms of efficiency, general satisfactory and serendipity.

What is Context

Context is a multifaceted concept and has been studied in various fields like computer science, cognitive science, psychology, etc. In each field, context has its own definition. To better understand how we can effectively use context in the mobile recommender systems, we need to find out the accurate definition of context in this field but not just borrow it from other areas.

According to Webster?s New Twentieth Century Dictionary(1980), context is the ?whole situation, background or environment relevant to some happening or personality.? This definition is too general and can not be effectively used in context-aware recommendation systems. In previous researches in context-aware computing, context is usually defined either through enumerating through examples of contextual information or through categorization.

Initially, context was defined as the location of the user, the identity of people near the user, the objects around and the changes in these elements [6]. After that, more factors were added to this definition. For example, Brown et al. [7] include the time of day, the season, and the temperature. Ryan et al. [8] add the physical and logical attributes of interest for a user. Dey et al. [11] include the user?s emotional and mental (focus-of-attention) status.

In paper [4], the author focuses on mobile context and divides mobile context into four categories:

- Physical context: This includes the physical status of or around the entities and can be the time, position, activity of the user, weather, light or temperature. For example, the weather of the day might affect the travel plan of the user.

- Social context: This includes the social status of the entities and can be the presence and role of other people around the user, or the relationship among different items. For example, a user might want to see different type of movies when s/he is with her/his boy/girlfriend and with her/his family. Also a system can recommend a user a scarf nearby that can be matched to the sweater the user just bought because of the matching relationship between the scarf and the sweater.

- Interaction media context: This includes the device the user is currently using, or the item the user is currently browsing or have bought.

- Modal context: This includes the states of mind of the user and can be the user?s purchasing goals, current mood, etc.

On the other hand, in paper [9], the author gives a more operational definition of context: Context is any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity, and state of people, groups, and computational and physical objects.

There is an argument among researchers whether context should only include information that is automatically obtained. Although, ideally, it is hoped that no manual input is required from the user which is the best from the user experience perspective. Many researches have also been done on automatic prediction of users? context. For example, in paper [13], the author tries to use time and location to infer user?s current activity. However, up till now, the techniques for automatic context detection is still immature and requires further research, we here opted for a more inclusive definition and accept context information that are either manually obtained and automatically obtained.

Context-aware Computing

Context-aware computing is a mobile computing paradigm in which systems can retrieve context either explicitly or implicitly and utilize contextual information to provide relevant information or services to the user, where relevancy depends on the user?s task [14, 1].

In summary, context can be used in three different ways in different applications [3]: The first way is to present information and services. It refers to the systems that present the context information directly to the user to help with the decision making of the users or the system propose appropriate selections of actions to the user. For example, in paper [15], a context-aware system was developed to present a choice of printers close to the user, but the system does not automatically book the printer or print the document for the user. The second way is to automatically execute a service. It refers to the systems that act or make decisions on behalf of the user based on the current context or context changes. For example, the Google Self-Driving Car use various detectors and sensors like laser to collect and generate data (e.g., maps) that is needed to allow the car to drive itself [16]. The third way is to attach context information for later retrieval. It refers to the systems that tag captured data with relevant context information so that the data can be retrieved in a different way using context.

In this thesis, we combines these three approaches. We present the detectable contextual information to the user so that they can decide which context factors is important in their decision making process. Then the system will automatically recommend clothes items based on users? configuration of context settings. Once the user purchases an item, the system will record the context and the bought item as a case so that it can be used for future recommendations.

Context and Reasoning Processes

After we know the importance and definition of context, we would like to know how the contextual information can be most effectively used by the system through studying how context affects people? reasoning processes. Paper [4] analyzes this from the psychological perspective. Psychologists Stanovich and West claims that there are two systems operating in the mind: System one operates subconsciously with little or no effort in solving the problem and system two operates allocates attention to the effortful mental activities that demand it. System1 always works under a certain context. It will generates a likely context if no context exists. For example, the music most recently played will affect your evaluation of the current music you are listening now. You may prefer music with similar style as last one because you are in a certain mood or you may prefer a music with different style because you want to try something new. In traditional recommender systems, recommendation quality is often measured by utility. If we look from the utility perspective, we can see that for the user, the quality of a recommender system depends on three factors [4]:

- Expected Utility: What you expect an experience will make you feel.
- Experienced Utility: The way an item (movie, travel, etc) makes you feel in the moment.
- Remembered Utility: Once you had an experience (e.g., a movie), future choice will be based on what you remember about that.

These three factors are produced, measured and utilized at different stages of the recommendation. The recommender systems predict recommendations based on the ?remembered? utility data. Users accept or reject based on expected utility. Users? experienced utility after consuming the item will be transferred into remembered utility and be used for the next round recommendation. So, in order to predict more precisely how likely a user may like an item, the system should know how to accurately measure these three types of utilities.

Researchers show that those utilities are affected and can be measured by some basic rules:

- Peak-end Rule

  D.Kahneman shows that what we remember about an experience is determined by how it felt when it was at its peak and how it felt when it ended. People rely on this summary later to remind how the experience felt and decide whether to have that experience again.

  Based on this rule, the recommender system should be able to detect and record users? feelings at the peak and at the end either explicitly or implicitly. For example, the system can ask for rating after the user have bought and tried the clothes as an end feeling and the system can record the purchasing of an item as the peak feeling. Those records can then be used as measurements of the utility of this item.

- Anchoring

  People measure the expected utility of an item by comparing it to some other items. For example, the shop will always put clothes with original price beside clothes with discount. Although most people will choose to buy those discount clothes, the clothes with original prices actually act as anchors against those discount clothes and make the prices of those discount clothes cheaper and more reasonable. So the recommender system should utilize this psychological rule and try to explain their recommendations in a more efficient way to increase the users? expected utility. For example, the system can put the original price of a clothes item when it is on sale at a more conspicuous place.

- Opportunity Cost

  Once you make a choice from several options, opportunity cost is the opportunities you need to give up if you choose a different option. Economists point out that the quality of any given option can not be assessed in isolation from its alternatives. For example, when recommending a discount clothes, the system can show some other similar clothes that do not have discounts to make the current recommendation be more appealing. Or the system can use some special icons to differentiate the items recommended for special contexts from other recommendations to decrease the opportunity cost like what was done in paper [5], the author mark some of the recommendations with an icon showing a small clock and a green arrow to show that these recommendations are especially suited for the current context.

Obtaining Contextual Information

The contextual information can be obtained in various ways including:

- Explicitly

  For contextual information that can not be detected automatically, the system can directly ask for it or elicit it from relevant people or other sources.

- Implicitly

  For contextual information that can be automatically detected, there is no need for the system to interact with the user or other contextual information for the data. The system can just implicitly gather the information without disturbing the user.

- Inferring

  The contextual information can also be inferred from other existing data known about the users and other entities. From the user experience perspective, it may help with enhancing the user experience because it reduce the manual input the user need to provide. However, this approach requires a predictive model which needs to be trained on appropriate data. Some research in this space already exists. In paper [12], the form of transportation can be inferred by the speed of the user using a decision tree (DT) followed by a first-order discrete Hidden Markov Model (DHMM). For clothes shopping recommender systems, we can also use user?s location (e.g., which area of the store the user is in) and movement to detect user?s current interest like whether or not they want buy male or female clothes, whether they want to buy sports clothes or formal clothes. However, because the indoor location detection technology is still not mature enough, we put this as future work in our thesis.

Relevance of contextual information

It is pointed out in paper [2] that not all contextual information is relevant or useful for recommendation purposes. It various for different applications and for different users and it is usually unclear which contextual factors are important and to what degree.

There are several approaches to determining the relevance of context factors. It can be either done manually, e.g., utilising the domain knowledge of the recommender system?s designer or the market expert, or automatically, e.g., analyse existing ratings data using approaches from machine learning, data mining and statistics during data preprocessing phase. In this paper, we follow the second approach mentioned above and adopted the methodology in paper [18] to quantitatively assess the relationship between contextual factors and users? intention to purchase. Traditionally, ratings data is collected in real situations which is time and energy consuming because enough data need to be collected in different context settings and it is not always the case that the ideal context situation will appear. In this methodology, contextual situations are simulated to more easily capture the data where users are asked to judge whether a contextual factor will affect their intention to purchase the item given a certain contextual condition. About how we determine the relevance of context factors using this methodology, please refer to section XX.

Incorporating Context into Recommender Systems

In paper [2], different approaches to incorporating contextual information into recommender systems are categorized into two groups:

- Recommendation via context-driven querying and search: This group refers to the systems that use contextual information and/or user?s specified interest to query or search a repository of resources (e.g., movies) and present the most appropriate result to the user. The contextual information is obtained directly from the user or the environment. The resources in the repository are tagged with contextual information while collecting them.
- Recommendation via contextual preference elicitation and estimation: This group refers to the systems that try to model and learn user preferences. Based on the data collected through observing user behaviors or from user?s preference feedback on previously recommended items, the system will try to model users? context-aware preferences and generate recommendations either through applying traditional recommendation approaches (e.g., context-based recommendation or collaborative filtering) or through applying various data analysis techniques from data mining or machine learning.

2.1.5 Mobile Recommender System

With the increasing of the computation power of mobile devices and the improvement of mobile network technologies, mobile phones are becoming a primary platform for information access [40]. This similar problem of information overload from PCs is also emerging in mobile devices and is becoming even more serious. Thus more and more researches have been on on deploying and developing recommender systems on mobile platforms so as to increase the usability of mobile systems providing personalized and more focused content and to solve the information overload problem.

However, existing recommendation approaches can not be directly applied to mobile platform because of two main reason. Firstly, mobile devices provide a different computing environment compared to PCs and has a lot of limitations and potential disadvantages. Mobile devices are of a smaller size compared to PCs and thus have limited screen size, computing power and data storage. Although users can read and understand the content displayed on small screens, it is still difficult for them to finish a recommendation session on them [40]. Users of mobile phones are usually less patient and be on the move, and thus have unstable network connection. They are usually less patient and tend to spend less time on mobile applications compared to PC softwares. So how to help users reach their goals in a short period of time efficiently is an important for mobile systems.

Secondly, mobile devices provide some extra characteristics and functionalities that can be exploited by recommendation approaches. The first characteristic is ?context-awareness?, i.e., the knowledge of user?s current context can be for easily detected and fetched either explicitly or implicitly. Nowadays, mobile devices can be equipped with multiple sensors. For example, motions sensors can measure acceleration forces and rotational forces along three axes. Air temperature and pressure can be measured by environmental sensors like barometers. User?s current outdoor location can be detected by GPS and even indoor locations can also be detected using, for example, iBeacon promoted by Apple [41]. The second characteristic is ubiquity [40], i.e., the ability to deliver the information and services to mobile users wherever they are, and whenever they need. This characteristic has put forward new challenges for mobile applications and services. Since users are interacting with the mobile applications more frequently and in more diverse set of contextual scenarios, the influence of the changes of the context conditions is also larger, hence the level of personalization, efficiency and accuracy is also higher.

As it can be seen, the constant changing of context conditions is a challenge for mobile recommender systems. On the other hand, mobile RSs can also benefit a lot from user?s current contextual information. Thus context-aware recommender system has become an important research field in recommender systems, especially for mobile RSs.

2.1.6 Baseline System

In paper [30], an offline shopping recommender system was developed on mobile platform utilizing conversation-based Active Learning strategy. Based on this system, we are going to build a context-aware mobile recommendation system using active learning strategies and see whether integrating of context-aware recommendation technique can help to improve the recommendation. So before we talk about the application developed in this thesis, we are going to first introduce the base system in this section.

This system is a conversation-based recommender system that actively select training points for user critiques so as to further adapt the recommendation according to the learned user preference from user?s critiques until a satisfactory item is found. The user can positively or negatively critique on the item features and the system will decide whether to find more items similar to the current critiqued item or to refocus and select more diverse set of items. It is argued that especially in an exploratory scenario such as going shopping without having a specific item in mind, the user don?t need to give a search query at the beginning of the recommendation. Thus a case-based recommendation approach using critiquing as feedback is adopted in the system.

Results show that the conversation-based Active Learning strategy is suitable for mobile situation and diversity-based information retrieval is preferred to only similarity-based retrieval method regarding prediction accuracy, user effort and the intention to return to the system.

Standard AL methods try to obtain ratings of representative training points from users so as to improve the predictive accuracy of the underlying recommender systems which are usually based on model-based approaches. Conversation-based AL methods, on the other hand, can quickly adapt to user?s current need by starting from a general set of recommendation and updating and narrowing down the scope of interest iteratively according to user?s feedback in each conversation cycle. It is especially suitable for users who do not have a clear preference at the beginning. Especially, regarding the clothes shopping scenario, it is unlikely that the user always wants to buy the same kind of items. Fashion changes, user?s taste also changes. At some point, the user will look for trousers, the next time for shirts. When the user is on a tight budget, she/he will prefer cheaper clothes, but some other day they may have more spendable cash available. Conversation-based system can quickly learn and adapt to those changes compared to a static model.

Since the system is used on a mobile platform, users are usually less patient than desktop users because of the limited interface space, the continuously moving state and the fact that they can be fully focus on the application because they also need to interact with real world objects around, case-based critiquing is used as a feedback in the conversational system, because in this case users don?t have to specify the exact preference at start which requires a lot of input and cognitive effort. Furthermore, the system does not rely on ratings from other users to infer recommendations as is done in collaborative filtering systems because data collection is time and resource consuming and also has privacy concerns. Case-base critiquing method is also suitable in this situation because it does not require any large-amount of pre-existing ratings.

Concerning all the factors above, an algorithm that can be used in conversation-based Active Learning systems using case-base critique as feedback by extending an existing algorithm called Adaptive Selection in paper [31].

Adaptive Selection (AS) is a conversation-based recommendation algorithm that uses user?s critiques as feedback between cycles. It differentiates itself from traditional similarity-based retrieval method by introducing a new diversity-enhancing technique so that a more diverse set of recommendations can be retrieved when the system detects that it is not homing in on the target region. If the system is heading in the right direction, the items most similar to the current query will be retrieved using similarity-based retrieval method.

AS detects whether the new recommendations are satisfying or not through, what McGinty and Smyth call, carrying the preference. In each cycle, the last critiqued item will be included in the new recommendations. It is argued that if the user keeps critiquing on the carried item, it is indicated that the new recommendations are not satisfying and the system should refocus and show more diverse set of items in the next cycle. Otherwise, the system should refine and show more similar items.

Algorithm 1 The bounded greedy selection of the Adaptive Selection algorithm by McGinty and Smyth.

function BoundedGreedySelection(q, CB, K, B)
    $CB\prime \leftarrow bkitemsinCBthataremostsimilartoq$
    $R \leftarrow \{\}$
    for $j = 1$ to k **do** Sort $CB\prime$ by $Quality(q, i, R)$ for each case $i \in CB\prime$
        $R \leftarrow R + First(CB\prime)$
        $CB\prime \leftarrow CB\prime - First(CB\prime)$
    end for
end function
$\alpha = 0.5$
$Quality(q, i, R) = \alpha Sim(q, I) + (1 - \alpha) RelDiv(i, R)$
$RelDiv(i, R) = \sum_{j=1...m}(1 - Sim(1, r_j))/m$ if $R! = ; = 1$ otherwise

The refocus function is relied on a quality measure which uses the bounded greedy selection algorithm (Figure 2.2) that ranks items according to their similarity to the query and their dissimilarity to already selected items. A further refinement is done on the quality measure by introducing a weighting factor . This factor allows to bias the quality metric in favor of either similarity to the target query or diversity among selected cases as can be seen in Equation xx.xx.

$$Quality(t, c, R) = \alpha * Similarity(t, c) + (1 - \alpha) * RelDiversity(c, R) \tag{2.4}$$

Paper [30] adopts the algorithm AS and modifies it for ease of use reasons by using a two-step critiquing process as the feedback method. Instead of determining whether to refocus or refine the recommendations based on whether or not the last critiqued item is critiqued again, in the new algorithm, users are asked to determine and tell the system whether she/he likes or dislikes an item before specifying what exactly those critiqued features are. For a detail explanation of the system please refer to paper [30].

2.1.7 Methodology for Developing CARS

There are four main issues that influence the successful design of CARSs [5]. The first issue is to find out the contextual information worth considering while generating recommendations. As was discussed in section xx.xx, not all contextual information is relevant or useful for recommendation purpose. After the context factors are selected, they can be integrated into the recommendation process either through context-driven querying and search or through contextual preference elicitation and estimation approach as was discussed in section xx.xx. Four both approaches, user?s in-context preferences knowledge need to be collected for either building knowledge bases or training predictive model. However this is a time and resource consuming process. Thus the second issue is to find an effective way for the collection of training data. After the recommendation approach is built, a complete recommendation system can be generated. Thus the third problem is to develop an efficient recommendation system using the given recommendation approach. The fourth problem is the interface design and visualization problem, including useful item description and explanations for the recommendation.

Paper [5] developed a methodology for solving these four issues and to support the development cycle for CARS. This methodology comprises four steps: determining which contexts are interesting to study; acquiring user ratings in specific contexts of interest; predicting ratings given a specific context; context-aware recommendation visualization and updating. In this thesis, we are going to adapt and apply this methodology to our system development. We are going to explain how each of these steps are implemented in this thesis.

3 Build the System

3.1 Acquiring Context Relevance

The first step of the methodology is to discover the relevance of the contextual factor to our current implementation domain (i.e. clothes shopping).

To adapt the recommendations to the user?s current contextual situation requires an understanding of the relationship between user preferences and contextual conditions. Thus it is proposed that explicit user ratings or any form of preferences should be given under several different contextual conditions. For instance, the user must rate a given clothing item when the temperature is hot, warm and cold which is quite time and resource consuming because user needs to only give ratings after they have experienced the context. Therefore, to reduce the risk of collecting data for unimportant context factors, an experiment needs to be first set up to determine which context factors are interesting to study.

We would like to know how the influence of each contextual factors change for different clothes types on user?s purchasing decisions and we also want to have a quantitative measure so that it can be used as weight in the similarity measurement in the following retrieving algorithm.

Consider all above reasons, we adopt the methodology developed in paper [18] to assess the context relevance.

This methodology is based on a web tool for acquiring context relevance judgements and a statistical data analysis method to quantitatively measure the influence of each contextual conditions on different clothes category. Following this methodology, we designed and developed a web survey. First, an initial set of contextual factors and conditions (values for the factors) were selected referring to some existing literatures about context-aware applications [5, 12, 19]. The selected contextual factors and conditions are listed in Table 1. Then, the clothes items were retrieved from zalando.co.uk. Especially, clothes of these brands are collected: Marc O'Polo, Tom Tailor, Esprit, S.Oliver, Benetton. Because these five brands are in the middle price category and are well known and generally acceptable by most people. Moreover, the types and number of clothes offered by these brands are similar to each other and are rich enough to cover most common clothes types. After the clothes were retrieved, they were aggregated into a relatively small list of categories so that the problem of data sparseness can be avoided. Totally, 14 categories were defined: tops, dresses, underwear, cardigans, trousers, coats, blouses, jackets, skirts, jeans, socks, swimwear, suits and shirts.

After the data was prepared, a simple web application was developed for acquiring the relevance of the selected contextual factors for the clothes categories (see Figure 1). In the web application, the user will be randomly given a clothes category and will be asked to imagine themselves being under a randomly chosen contextual condition and then choose the influence of the selected contextual condition on their intention to buy the selected type of clothes. As an example of the questions posed to the user consider the situation depicted in Figure 1. Here we first ask the user to imagine a typical shopping scenario: ?Imagine that you are in Munich and you are doing offline shopping for clothes. You are thinking about buying Skirts.? Then the user is asked to select the influence (i.e., positive, no effect, negative) of the three randomly chosen contextual conditions on their decision to buy the clothes. As an example of a contextual condition: ?Imagine that the weather is cloudy.? Every user was requested to interact with at least 10 of these pages (as in Figure 1).

38 participants took part in this web survey. Overall 1190 responses were given to one of the questions shown in Figure 1. Because no pre-knowledge is known whether certain context conditions are more likely to influence user?s decision, we sampled the value of clothes categories and contextual conditions using uniform distribution so that all possible values can be reached with equal opportunities.

3.1.1 Analysis of Context Relevance

The goal of this web survey in this thesis is to find out quantitatively how the context factors influence user decisions whether or not to buy clothes from different categories.

The web survey delivered samples for the distribution $P(I|C_i, T)$ where I(Influence) is the response variable taking one of the three values: positive, negative, or no effect, T is a clothes category (e.g., tops, skirts), and $C_1, ...C_N$ are the context factors that may or may not influence the user decision. This distribution models the influence of the context factors on the user?s decision considering different clothes categories.

The spread of a categorical variable $X = x_1, ...x_n$ can be measured by looking at the entropy of the random variable. If $P(X = x_i) = \pi_i$, the entropy of X is:

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

This measurement of the spread can be used to estimate the association between variable X_1: user?s intention to buy a certain item (i.e., positive, negative or no effect) and variable X_2: one of the current context factor (e.g., current budget). Informally, if the influence of the context factor is strong, then the spread of variable X_1will be reduced if X_2 is known, and it is weak if the spread of X_1 remains unchanged even if X_2 is known and this association can be formally defined as [18]:

$$U = \frac{E(X_1) - \varepsilon(E(X_1|X_2))}{E(X_1)}$$

where $E(X_1) - \varepsilon(E(X_1|X_2))$ is the difference between the spread of X_1 and the expected spread of X_2 which measures the influence of the context factor to user?s decision, where $\varepsilon(X)$ denotes the expected value of the random variable X. As the spread of $(X_1|X_2)$ is the same as the spread of X_1. We computed U for all context factors and clothes categories and the ordered factors in descending order of U for each clothes category can be seen in Appendix of this paper.

TODO Some simple analysis of the result

3.2 The Proposed Approach

In this section, a recommendation approach is built, the goal of which is to integrate the contextual information into the recommendation process and recommend items that might be of interest to user under a specific context situation.

To integrate the contextual information into the recommender system, we are going to adopt the context-driven querying and search approach. As was introduced in section xx.xx, 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. The repository usually contains resources that are tagged with contextual information while collecting them. Corresponding to this approach, we use case-based recommendation technique to realize it. As was discussed in section xx.xx, case-based recommendation technique is a branch of knowledge-based recommendation. Compared to collaborative filtering and content-based technique, case-based recommendation has no cold-start problem because it can rely on the case base (or knowledge base) for initial recommendation. This case base can be set up using expert experience quickly and does not require pre-training like what is done in paper [5] for model-based approach.

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. For example, the user may ask for recommendation of clothes for work (condition one) when the weather is warm (condition two). For the recommendation, the user first submit the contextual information as the query. Then the system will search in the case base and select the cases with the most similar context situation and then recommend the items or items similar to the items contained in the cases to the user.

On the other hand, knowledge-based recommender system has the disadvantages of static suggestion ability because the knowledge-base is usually preset by domain expert and barely changes. Also usually the domain expert and the knowledge engineer are not the same person, the communication cost requires an efficient way for knowledge engineering. Based on those considerations, we extend the case-based recommendation by integrating collaborative filtering approach for case base (knowledge base) setup so that all the application users will play the expert role and their purchased items together with the contextual information will be added to the case base as a new case for future recommendation. Although the collaborative filtering approach is used, the correlation between users is performed at the session level (i.e., each submitted case is independent of itself and will not be related to the user who submit the case). Thus no user identification is required and a considerable amount of example data is not needed for each single user in order to deliver effective recommendations.[26])

3.3 Towards Case-Based Recommender Systems

As was introduced in section xx.xx, Case-Based Recommender Systems (CBRSs) apply case-based reasoning (CBR) methodology to solve recommendation problem by re-using or adapting past recommendation solution stored in past similar cases. In the CBRSs framework introduced in paper [22], a typical recommendation process is composed of six steps: retrieval, reuse, revise, review, retain and iterate. We are going to apply this framework to the building of our context-aware CBRS.

Input: To get a list of recommendation, the user will first submit a query of current contextual information. A 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 recommendation. For example, if a user is a budget buyer and want to buy clothes for sports purpose when the temperature is hot and the user wants to find clothing items sold in stores that will be still open in the next 30 minutes within 2000 meters, the query will be structured as follows:

$$\begin{aligned} query = \{((distance \leq 2000m) \wedge (timeopen = now + 30min)), \\ (budget(budgetbuyer), intent(sports), temperature(hot))\} \end{aligned} \tag{3.1}$$

Retrieval: After the user submit a context query, contextual factors such as budget, intent etc. will be used to find and rank cases with similar context. The definition of similarity and the similarity assessment algorithm will be introduced in section xx.xx.

Reuse: In the final recommendation, nine items in the cases will be recommended to the user. However those items will not be ranked only according to the similarity of the cases to the current context. As for initial recommendation of conversational recommendation system in an exploratory mobile scenario, diversity is an important consideration to ensure the coverage of the current scope of candidate items. Thus we extend the bounded greedy selection algorithm to select the cases with the most diverse set of items among the retrieved most similar cases.

Revise: Before the items are recommended to the user, logical constraints such as distance (e.g. find clothes within 2000 meters) or open time (e.g., shops still open in the next 30 minutes) will be used to check the availability of those items. If for example a recommended item is too far away , then other similar items will be recommended instead.

Review & Iterate: After the initial recommendations are presented, users can update the recommendations iteratively through critiquing directly on item features. This is also called conversation-based Active Learning strategy and has been explained in detail in section xx.xx.

Retain: Finally, when the user selects and purchases an item, the time together with the current context situation will be stored as a new case in the case base.

3.4 Case Model

The Case Base is made of made of two components: item bought I and context situation C:

$$CB = I \times C \tag{3.2}$$

Each case $c = (i, e) \in CB$ in the case base will be consist of two sub-elements i, e which are instances of the spaces I, C respectively. As was introduced before, the cases will not be correlated with the user who submits it, thus user model is not contained in the case in our system. A case is built during a human/machine interaction [26]. In our system, a case is created when the user purchases the item. According to Peak-end Rule introduced in section xx.xx, how a user feels about an experience is highly influenced by the end of the experience. We assume here that users give high rates for the items they buy. Since the case is created when the user bought an item, we here get rid of the evaluation model as well in the case base. In the following we will introduce these two components in detail.

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 value that the user wishes to be considered in recommendation and a feature value vector of context factors and their corresponding factor importance weight. The factor importance weight reflect the level of influence of the context factors for the recommendation of the clothing item contained in the same case. They are determined by the clothing type of the clothing item and has been calculated using experiment in

Section xx.xx. For a full list of the factor importance weights for different clothing type please refer to Table xx.xx. The main context factors are: distance, day of the week, temperature, time available, transport, weather, time of the day, crowdedness, intent of purchasing, companion, season and budget. For a detail list of the context factors and their values, please refer to Table XX. For a typical example, if a user is a budget buyer and is looking for clothes for sports when the temperature is hot, the context situation can be structured as follows:

$$\begin{aligned} context_{attributes} = \{ & (budget(budgetbuyer), intent(sports), temperature(hot)), \\ & ((w_{budget}(0.7), w_{intent}(0.6), w_{temperature}(0.9))\} \end{aligned} \tag{3.3}$$

I is the data structure that describes the clothing item bought by the user. It is represented as a feature value weight vector (Equation xx.xx). It is directly borrowed from the base system introduced in section xx.xx.

Through this case model, knowledge about what kind of items users buy in a certain context situation can be obtained. To provide recommendation, cases with context situation similar to the current user can be retrieved and the items contained in those cases can be used directly for recommendation. They can also be used as reference items to find other similar items to recommend. Thus we are going to show how the similarity between current context and retrieved cases and similarity between items are calculated.

3.5 Similarity Assessment

To get the similarity between current context and retrieved cases, we borrow the Euclidean Overlap Metric (HEOM) [26, 34]:

$$heom(x,y) = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i}} \sqrt{\sum_{i=1}^{n} w_i d_i(x_i, y_i)^2} \tag{3.4}$$

where:

TODO the equation

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. This metric measures the distance between two vectors. Thus the further away two vectors, the more similar they are. We modified this metric so that it can be applied to our system.

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. First, for all the context factors submitted in the user?

query (currentContext), we will calculate the similarity to the target context factors (targetContext) in the cases. In some cases, the targetContext may not contain context factors that are specified by the user (e.g., the user enables the context factor shopping intent and budget, but the targetContext only contains budget). In some other cases, the currentContext may not contain some context factors contained in the targetContext. So when computing similarities, we use user specified context factors as base and only consider the similarities between context factors specified by the user. The context factors contained in the targetContext but not in the currentContext will be ignored, because the user chooses to ignore those factors. If the targetContext does not contain the context factors specified in currentContext, the similarity will be added by $1 * w_i$. (w_i here corresponds to the feature factor weight).

The simplified similarity metric is displayed as follows:

Algorithm 2 The simplified similarity metric

function GETSIMILARITY($query, case$)
  $targetContext \leftarrow getCaseContext(case)$
  $currentContext \leftarrow getQueryContext(query)$
  for all $contextfactorsdefinedintargetConetxt$ **do**
    if the $targetContext$ contains the current context factor in $currentContext$ **then**
      $sim \leftarrow sim + factorSimilarity(\pi_f(targetContext), \pi_f(currentContext))$
      $weight \leftarrow weight + getWeight(case, \pi_f(targetContext))$
    else
      $sim \leftarrow sim + 1 * getWeight(case, \pi_f(targetContext))$
      $weight \leftarrow weight + getWeight(case, \pi_f(targetContext))$
    end if
  end for
end function

After the similarities of the cases retrieved are calculated, they will be first ranked according to the calculated similarity, the most similar the first. Then we use the bounded greedy selection algorithm to select and rank the cases based on the diversity of items contained in those cases.

3.6 Explanation Generation

The learned factor importance weights contained in each case will be used for generating explanations for the recommendations. Analyzing the learned importance weights one can generate explanation based on the values of these parameters. More specifically, given a case that includes an item i and a context situation c in which a set of context factors $(c_1, ..., c_k)$ as well as the corresponding factor weights $(w_1, ..., w_k)$ are specified, we first find out the set of context factors $(c\prime_1, ..., c\prime_j)$ that overlap with the context factors specified in user?s query as well as the corresponding factors weights $(w\prime_1, ..., w\prime_j)$. Then we identify a fixed number of the context factors f, g, h (in this thesis, at most three context factors will be identified), among those overlapped context factors, with the highest importance weights $w\prime_f, w\prime_g, w\prime_h$. If the number of overlapped context factors is smaller than the fixed number, then the whole set of overlapped context factors will be used. After those identified context factors is ranked in descending order based on their corresponding weights, they will be used to generate a positive explanation for recommending item i. For example, if item i (e.g., a dress) is recommended in the contextual situation ?the shopping purpose is for party, the user is a budget buyer and the temperature is hot? and the overlapped contextual conditions are ?the shopping purpose is for party? and ?the user is a budget buyer?, we observe factor ?budget? has a higher importance than ?shopping purpose?, so we explain that the dress is recommended because ?other user bought similar clothes when the she/he is a budget buyer and the shopping purpose is for party?.

3.7 Interaction and Interface Design

In this section we illustrate the main features of Shopper, a mobile context-aware recommender system. Shopper is a Android application built based on a personalized recommender system using conversion-based active learning strategy developed in paper [30]. Shopper integrates contextual information using case-based recommendation approach into this system so that users can obtain recommendations adapted to the recommendation context. The detail of the case-based recommendation approach has been discussed before in section xx.xx. To get an initial set of recommendations, the user makes a recommendation request specifying contextual conditions and then a list of clothing items (including pictures and descriptions) will be returned. Those recommendations are obtained through searching for items bought by other users under similar items. Then the user can critique and update the recommendations iteratively to get the most satisfying item. We will not describe this system illustrating a typical interaction.

In the initial step of the interaction with Shopper the user normally sets the current shopping context. Figure xx shows the GUI for enabling and setting the values of the selected contextual factors. It is built using Android's Preference APIs so that the settings interface will be consistent with the user experience in other Android apps (including the system setting). Here we can see the user can switch on/off some of these factors, e.g., ?Time of the day? or ?Weather? using checkbox. When these factors are switched on the recommender system will take into account their current values (conditions) by querying a third party service. For example, to get the current weather and temperature, the system will query the Yahoo weather API and parse the returned XML file for the target information. For some other context factors, e.g., ?Budget? or ?Intent?, the user can switch them on or off by selecting the ?Off? option in a pop up dialog. When those factors are switched on, the user need to provide the value manually by selecting among the options in a pop up dialog as in Figure xx (right). After the user close the dialog, the selected value will be shown as subtitle under the corresponding item line so that the user can easily have a clear view of what are enabled and selected. The full set of contextual factors is the same as in the web application described earlier and their values could be found in Table xx. The contextual conditions: distance, day of the week, temperature, weather, time of the day, crowdedness and season are automatically obtained from third party services. The remaining contextual conditions, if the user has enabled them, must be entered manually by the user.

After the user has enabled some contextual factors and provided the values for them, the system can be requested for recommendations. A short number of recommendations (nine in this system) will be represented to the user as depicted in Figure xx. If the user is interested in any of the item, she/he can click on the picture and see the detail page of the item as depicted in Figure xx. In the detail view, the user can see an explanation of the reason why this item is recommended to the user because we believe explanation will boost the transparency and user?s trust to the system. A typical explanation can be ?Other user bought similar clothes when they are feeling like a party animal, they are a budget buyer, it is weekend.? Those are the contextual conditions that are most influential for recommending the current clothes item (as was explained in previous section).

The initial consideration is to put the explanation in the main view so that they can be more directly observed by the user. However, it can be seen that there is already critiques explanation in the main view. Considering the limited space of mobile devices, the user experience will be reduced if too much text is displayed together. Moreover, it might be the case that each item has a different reason for being recommended. For example, if a user requires for recommendation of clothing item for context situation ?budget buyer, for work and for cold temperature?. One case satisfying contextual condition ?budget buyer? and another case satisfying contextual condition ?for work and for cold temperature? can all be recommended to the user in this case. Thus it is better the put the explanation in the detail page of each item separately.

In the detail view, the user will also see an explanation of the location of the clothes as well as a map view of the location of the clothes relative to user?s current location as depicted in Figure xx. For users who only want to find clothes nearby, it is an important factor for purchasing decision. If the user is not satisfied with the recommended clothes, she/he can critique on the item features to iteratively update the recommendations. This step corresponds to the review and iterate step in case-based recommendation as was discussed in section xx.xx. It relies on the main functions of the system in paper [30].

If the user finds the ideal item, she/he can enter the detail view and click on the button ?select and finish? (Figure xx.xx). Then the selected item together with the current contextual situation will be recorded as a new case in the system case base.

4 Evaluation

4.1 Motivation and Goals

Initially, recommender systems are usually evaluated based on their prediction accuracy - how accurately they can predict the items that the users are most interested in. However, it is now widely agreed that accuracy is an important but not unique measurement for the quality recommendation systems, because the user is not always looking for the best prediction of their tastes. Especially in cases where the user do not have specific preferences in mind and is still in the exploration stage, users may value more the diversity and novelty of the recommendation. Moreover, if the RS is built on a mobile system, user is usually less patient when using the system, decision effort that is needed in finishing a task is also an influential factor for the quality of the recommender system. Thus, besides the evaluation of prediction accuracy, we will also measure user?s decision effort and user?s general satisfaction of the system. Especially concerning the context-aware feature, whether the users are well aware of the benefits of context and the corresponding context-aware explanation will also be measured.

There are three different types of experiments: offline, user studies and online experiments [35]. Often it is easiest to perform offline experiments because no user is involved. Through using existing data sets and a protocol that models user behavior, recommender system performance such as prediction accuracy can be easily measured at low cost. However, because the narrow set of properties it can measure, and the fact that it is difficult to create a reliable simulation of user interactions with the system [35], this approach is not appropriate for our evaluation. A more expensive option is a user study, where a small number of users are asked to perform a set of tasks using the system. A set of raw metrics will be measured during the task for future statistical analysis and users usually need to answer a set of questions about their feelings of the system. This type of experiment provides a controllable experiment environment so that it can be adapted and used to measure different properties. It is suitable for recommendation approaches that rely on the interaction of users with the system. Finally, for realistic recommender systems, online evaluation can be used where the performances of real users are evaluated when they are using the system and is not aware of the ongoing experiment. However this approach can not be adapted to easily to specific evaluation requirements. Based on all the discussions above, we are going to use the user study experiment for the evaluation.

Two variants of the system will be tested. The baseline system is presented in section xx.xx. The other one is introduced in section xx.xx. To test all candidates within subjects methods will be used, where each subject tests a set of candidates on different tasks [36] so that the two system variants can be more reasonably compared to each other. However, the same people testing both variants can introduce a bias due to for example a learning effect (e.g., the user may spend less time on the second tested variant because she/he is more familiar with the interface when using the first variant.). To avoid such bias, the order of variants to test by the user should be randomized and enough time should be left to let the user get familiar with the system first. Moreover, the case base (knowledge base) is kept fixed in the evaluation so that the recommendations are all based on the same case base.

When drawing conclusions from experiments, it is hoped that the conclusions can be applied to general cases rather only to the context of the experiments. To increase the probability of generalization of the results, firstly, the experiment participants should represent as closely as possible the true population of users of the real system. In our evaluation, people of various ages (from 21 to 56) and occupations (e.g., freelance, student, teacher, software engineer) are invited to participate in the user study so that the test result can be as much generalized as possible. It was aimed to look for around 30 participants so that the number of sample data can be big enough to be evaluated using a paired t-test. To not overwhelm and exhaust candidates, the experiment will be controlled in 10 to 15 minutes.

4.2 Data Set

4.2.1 Data Set for Clothes Items

In this evaluation, data set retrieved in section xx.xx will be used as sample data. The extracted data is from online store Zalando as was introduced before and may not be actually available offline here in Germany. However for evaluation purpose, they will be randomly assigned to a list of offline stores available in Germany.

After the raw data is retrieved, it is preprocessed to get rid of irrelevant categories such as "Home" and "Child". The depth of the initial category tree is five and we only keep the category information up to depth 2 and we merge some of categories that are similar to each other but have different names to avoid sparsity problem. For example, "Lingeire & Nightwear" in women?s clothes category is merged with "Underwear" in men?s category. So finally, 14 categories are defined: tops, dresses, underwear, cardigans, trousers, coats, blouses, jackets, skirts, jeans, socks, swimwear, suits and shirts. Then for each clothing item, irrelevant information is removed and the following information is kept:

- a numeric identifier (id),
- the clothes name,
- the price (in Euro, stored unit-less),
- one of 34 colors,
- one of 12 brands (including the sub brands such as "QS by s.Oliver"),
- the sex,
- one of 14 types of clothes,
- the link to an image of the item

The result set contains 3920 items in total with 2238 clothes items for female sex and 1682 clothes items for male sex. For each clothing type there are between ???

4.2.2 Data Set for Context Case Base

As was mentioned in previous section, the case-based context-aware recommender system developed in this thesis recommend based on a case base which can be built up through collaborative approach, as well as expert-driven approach. For the evaluation in this thesis, expert-driven approach was used to first set up a starting case base. Since the users were asked to make purchasing decisions while imagining themselves being in a certain context scenario, the starting case base should be as various as possible for the corresponding scenario. Take buying clothes for sports purpose for example, users? potential choices may vary from sports trousers, sports shirts to swimming suits, yoga pants. Apparently, it can be difficult to find enough people with choices various enough for the case base in a short period of time, it is easier and more efficient to use expert-driven knowledge acquisition approach.

Clothes selection was done for each of the five predefined context scenarios that would be used in the evaluation task, for which both common sense and Zalando website were used as the main resource. For example, according to the common sense, thick clothes such as jacket, sweater, long trousers, coat will be selected for cold weather, clothes such as dresses will be selected for party purpose. On the other hand, in the Find your style page in Zalando website, different styles such as going out, romantic, casual, business are defined and they were also used as reference for the selection of clothes for the case base. In total, ten cases were added for each context scenario with five cases for male and five cases for female.

4.3 Test Setup

In the baseline system, as was introduced in section xx.xx, the system recommends without initial preference from the user. To optimize the initial recommendation quality, a set of items are selected using a diversity-enhanced technique to ensure the coverage of the presented items. Compared to that, the system developed in this thesis tries to improve the the initial recommendation through integrating complex contextual information into the recommendation algorithm. So the baseline system is a simplified version of the developed variant. It does not consider contextual information during recommendation and the context setting UI in Figure xx is removed. However for both systems, the user can critique on items features and iteratively update the recommendations according to their personal interest.

The test hardware is a 3.7 inch 480 x 854 resolution Android smartphone (Motorola DEFY) running the Android operating system of version 2.3.7.

4.3.1 Testing Framework

In this evaluation, recommendation quality is mainly qualitatively measured with the standard questionnaire mentioned before. Quantitative information is also collected to better interpret the result. The measured data is divided into four areas: prediction accuracy, decision effort, explanation benefits and context benefits.

Prediction Accuracy

Prediction accuracy is by far the most discussed and the most important property for recommender system evaluation. It is based on a basic assumption that a system that provides more accurate predictions will be preferred by the user [35]. It measures the ability a system can help users find the items they like. In this thesis, this property is measured qualitatively by asking the user which system suggest more appropriate clothes. The appropriateness here reflect the accuracy of the recommendations.

Decision Effort

People usually want a recommender system with high prediction accuracy and low decision effort. However, strategies yielding more accurate choice are often more effortful, and easy strategies can lead to lower levels of accuracy [38]. Especially in mobile exploratory scenario, a recommender system that is accurate but takes long time or a lot of cognitive effort to reach is also not useful. Thus in this evaluation, the decision effort is measured using both qualitative and quantitative methods.

Quantitative measurements. The decision effort can be quantitatively measured by the task time and the interaction effort. Task time is the time a user needs to find and select an ideal item after she/he is presented the initial recommendation. Interaction effort here can be represented by the number of critiquing cycles. One critiquing cycle is counted when the user issue a critique on the item features, triggering the selection of new recommendations.

Qualitative measurements. The decision effort is also measured quantitatively by asking the user to rate the easiness of finding the information she/he needs using the system and asking the user the effectiveness of the system in helping her/him to complete the scenario. The scenario here means giving a context scenario, finding an ideal item to buy.

Context Benefits

In spite of the goal of improving the quality of the recommendation, the integration of contextual information into the system also provides a new way for user to personalize their recommendation. To study whether or not the user is interfered by the extra step of context setting and whether or not the user is aware of the benefits of context-aware recommendation, the user is asked does she/he understand the benefits of using contextual conditions.

Explanation Benefits

From the user?s perspective, explanation is an important experience because it helps the user understand how the recommender system work. Explanation is supposed to have many benefits including improving the transparency of the system, increasing the trust of the user to the system, persuading user to try or buy the recommended item etc. So in this evaluation, users are also studied to see whether they are aware of the explanation benefits through asking them whether they think the contextual explanations are useful. The clearness of the contextual explanations are measured qualitatively by asking the user whether she/he is satisfied with the provided contextual explanations and whether the contextual explanations provided by the system are clear.

4.3.2 Testing Procedure

The actual testing procedure used in the evaluation was structured as follows:

1. To introduce the application to the user. Because the users are of different knowledge backgrounds, not all of them are familiar with terms like ?recommender system? or ?context?. Since most of them have online shopping experience, to let the users quickly learn the concept of recommender system and the core functions of the

application, we use website like Amazon as a real life example to help the user gain a quick feeling of what our application is about.

After the users know the basic idea of the application, we will explain to the user how to use it. Nowadays, mobile applications will show a quick graphic instruction to tell the user how to use the system when they are first launched. Here a manual instruction is given with the same purpose.

2. The user is asked to play around with the application. Before the start of the evaluation task, the user is asked to play around with both variants of the system and ask questions whenever they are confused. They will be asked to finish a simple task (i.e., select an ideal item using the current system). Usually, more questions will come out when they are trying to finish this simple task. In this way, it can be made sure that during the formal study, the user can focus on the task itself but not figuring out how to use the app.

3. Explain the task setting to the user. There are two variants of the system as was introduced before. While testing with both variants, the user will be asked to select an item they are most satisfied with while imagining themselves being in a context scenario that are randomly selected by the system from a set of five pre-created context scenarios. A typical context description can be:

Imagine that you want to buy clothes for daily wear,you are a budget buyer and the temperature is cold.You don't care the exact location of the clothes.

For each context, usually four factors will be included. To see the full list of those scenarios, please refer to Appendix xx. The user will be first explained to the concept of context and then be shown to the descriptions of the context scenarios that may be randomly selected during the user study. In this way, the user will get a full view of the study, feel more in control of the study and feel more confident in their actions.

The amount of advantages or disadvantages of context-aware recommender system compared to non context-aware recommender system may vary among different pre-created context scenarios. To avoid this bias, for each user study, the two system variants are tested using the same randomly chosen context scenario so that they can be compared under the same controlled situation.

One thing that needs to be pointed out is that because in the formal study, when users are using the context-aware variant developed in this thesis, they don?t need to configure the context settings by themselves and the system will be automatically configured to be in the chosen context scenario. Because for pre-defined context scenarios, the context conditions such as the weather or the temperature can be different from the current one, thus must be programmatically set and can not be automatically retrieved from third party services. In some cases, the user will forget the existence of the context setting step. To avoid this situation, users will especially be asked to use the context setting function and see the reaction of the system (updating of the item list) according to the change of their context settings in the previous step and make sure that they understand how the context-aware system work. They will also be explained carefully the reason why they don?t need to configure the context in real user studies.

4. Conduct the task. Once the user understands the task mentioned above, a variant of the application will be set up and handed back to the user. After the user selects one item, they will be given to another variant and do the task again. To reduce the effect of learning curve, the order of the first variants to give will be randomly selected. Users are also encouraged to give oral feedback during the task.

At first the testing procedure was designed to ask the user to test each variant of the system twice, each time with a different context scenario, so that the user can be aware of situations when context changes and different reactions of these two variants. However, consider that this procedure design is too time consuming, with the goal of not overwhelming the user, this design was given up and the current procedure design is adopted.

5. Finish the survey. After the user tests on both variants, she/he will be asked to fill out a survey. The survey is created online using google form so that the user can fill it out privately. In this way, the influence of the interviewer will be reduced to the most. The survey includes the following statements:

Q1: It was easy to find the information I needed. Q2: The system is effective in helping me to complete the scenario. Q3: I like using this system Q4: I understood the benefit of using the contextual conditions. Q5: I am satisfied with the provided contextual explanations. Q6: I believe that the contextual explanations are useful. Q7: The contextual explanations provided by this system are clear. Q8: Which system do you prefer? Q9: Which system suggest more appropriate clothes?

Statements 4-7 were provided only for the context-aware version. Statements 4-7 were asked for both variants. At the end of the survey, users were asked which system they preferred (Q8) and which system suggested more appropriate points of interest (Q9).The user could express a level of agreement to the statement ranging from 1 (strongly disagree) to 5 (strongly agree). These questions were extracted and adapted to our evaluation from the IBM Computer System Usability Questionnaire [37].

4.4 Results

In this section, the data collected in the user study will be analyzed for each property measurement.

4.4.1 Participants

For the study, participants of various age, religion, nationality, knowledge background and current profession were looked for. Overall a number of 23 people participated, among which 17 people were students as well as employees in academia in fields including computer science, electronic engineering, finance, economics, and other 6 people were employees and freelancers from outside of the university. The average age was 27, with a maximum of 56 and a minimum of 21.

4.4.2 Overview

The measurements for decision effort and system preference for both two systems (CW denoting the variant using context-aware recommendation, NCW the one recommending without contextual information) are shown in Table xx.xx. Next to the mean is the standard deviation and the last column denoting the p-value of a one-tail paired t-test with 22 degrees of freedom (23 participants - 1).

In the following sections, a detailed discussion of the measurements of the four properties (prediction accuracy, decision effort, explanation benefits and context benefits), the general system preference as well as the informal feedback will be given.

Table 4.1: Overview of the result.

		CW mean	stdev	NCW mean	stdev	p value
Decision Effort	It was easy to find the information I needed.	3.74	0.92	3.61	1.16	0.3
Decision Effort	The system is effective in helping me to complete the scenario.	4.09	0.79	3.74	0.92	0.029
Decision Effort	Critiquing cycle	2.83	2.46	3.43	2.35	0.171
Decision Effort	Completion time	122.91	77.67	117.52	73	0.405
System Preference	I like using this system	4.04	0.77	3.39	0.94	0.004

4.4.3 Prediction Accuracy

After the users tried both systems, they were asked which system provided more appropriate clothes (Q9). The majority (87%, 20 people) selected the context-aware variant (CW). Among the three people who selected non context-aware variant, two of them still preferred the context-aware when they were asked to selected the preferred system.

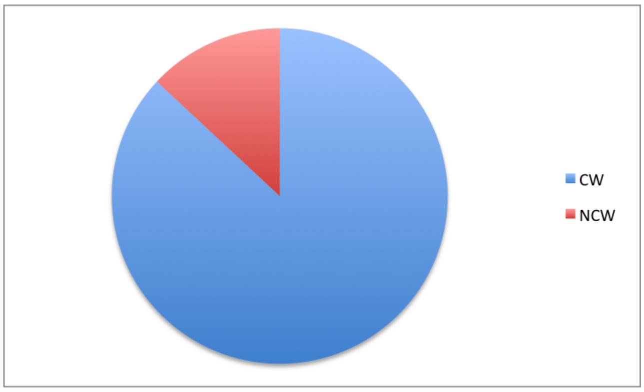

Figure 4.1: Prediction accuracy in terms of which variant provided more appropriate clothes.

4.4.4 Decision Efficiency

The quantitative measurements for decision effort are broken down into the number of critiquing cycles and the time it took to complete a recommendation session.

Critiquing Cycles

Figure xx.xx shows the box plot of critiquing cycles for CW and NCW. It can be seen that the box of CW is obviously lower than the box of NCW, which suggests a difference between CW and NCW in critiquing cycles with cycles of CW being generally lower than the ones of NCW. The mean of critiquing cycles for CW (2.83 cycles) is also lower than the one for NCW (3.43 cycles) in Table xx.xx but the difference is not significant (p=0.171). It can be also found that the box of NCW is comparatively shorter than the box of CW which suggests that some of the critiquing cycles of CW varies more than the ones of NCW. This can also be reflected from the standard deviations of critiquing cycles of these two variants in Table xx.xx (2.46 for CW compared to 2.35 for NCW).

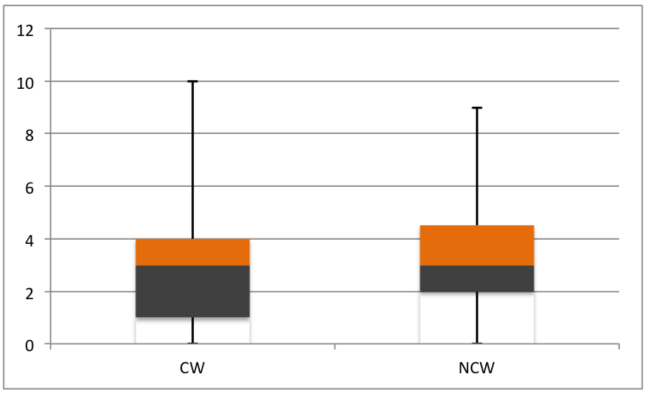

Figure 4.2: Decision efficiency in terms of critiquing cycles.

Critiquing Time

Figure xx.xx shows the box plot of critiquing time for CW and NCW. It can be seen that the medium of CW (109 seconds) is larger than the one of NCW (81 seconds), but the the box length of CW is comparatively shorter than the one of NCW, which suggests that the majority of the critiquing time of CW is more stable and is neither too long nor too short. For both two variants, the maximum (302 seconds for CW and 297 seconds for NCW) and minimum (26 seconds for CW and 18 seconds for NCW) critiquing time are almost the same. From Table xx.xx, it can be seen that from a general view, the critiquing time of CW varies more than that of NCW. NCW slightly beats CW in critiquing time when looking at the average time in Table xx.xx, however, NCW was found to not be significantly better.

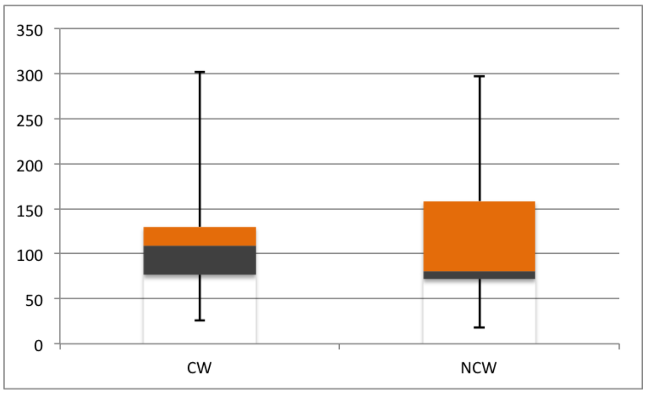

Figure 4.3: Decision efficiency in terms of critiquing time.

The qualitative measurements of decision efficiency are broken down into the rate for the ease of use of the system and the rate for the effectiveness of the system from the users.

Ease of Use

To measure the ease of use of the system, the user was asked was it easy to find the information she/he needed (Q1). Whether or not useful information can be easily found depends on both the design of the recommendation approach and the design of the interface. When looking at the mean of the rate of this question in Table xx.xx, CW slightly beats NCW (3.74 against 3.61). However the difference is not significant enough (p=0.3). The standard deviation of CW (0.92) is smaller than the one of NCW (1.16), which suggests that users have a higher level of agreement on the mean rate of CW.

When looking at Figure xx.xx. It can be seen that more users gave score five for NCW than for CW (seven people for NCW and six people for CW). However, four people also gave negative rates for NCW (value 2), which did not appear for CW. Quite a number of people (9 people) hold a neutral view towards NCW compared to CW (6 people).

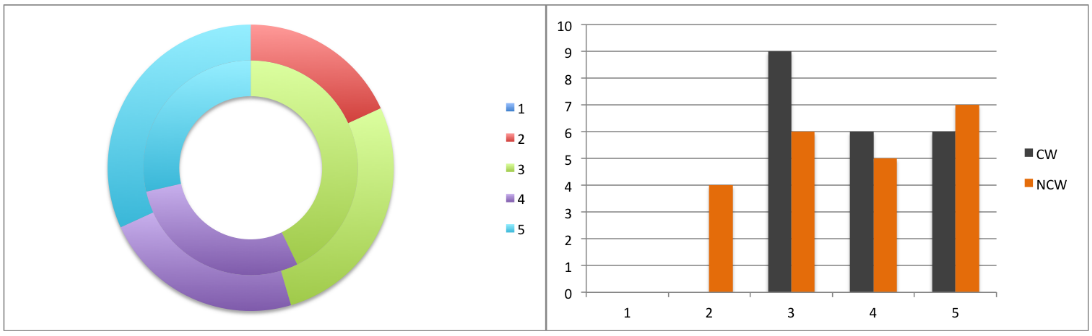

Figure 4.4: Decision efficiency in terms of distribution of ratings for ease of use on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right

Effectiveness

To measure the effectiveness of the system, the user was asked the question was the system effective in helping her/him to complete the scenario (Q2). In the evaluation, the user was given a context scenario and was asked to imagine her/himself being in the context scenario while using the system. With this context-aware scenario setting, CW is rated significantly better than NCW ($p=0.029 < 0.1$). Some users mentioned that they found the context setting function quite useful during the test. The average rate of CW (4.09) is also higher than that of NCW (3.74). On the other hand, the standard deviation of CW (0.79) is smaller than that of NCW which suggests that users had a higher degree of agreement towards the rating of CW than NCW.

If we look at Figure xx.xx, CW collected twice the number of positive (value 4 on the Likert scale) ratings and three quarters of the number of neutral ratings (value 3) compared to NCW variant. Also NCW received a negative ratings (value 2), while CW obtained a shared positive view.

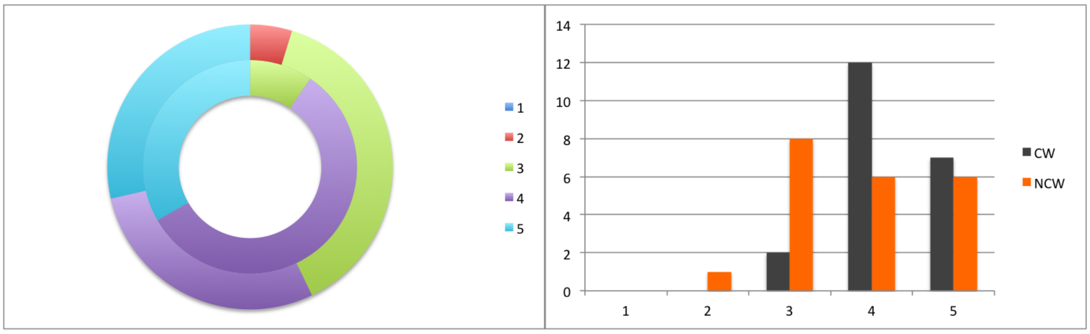

Figure 4.5: Decision efficiency in terms of distribution of ratings for effectiveness on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right

4.4.5 Context Benefits

To investigate whether users intuitively understand the benefits of using contextual information for recommendation but not seeing it as an obstruction, users were asked whether they understood the benefits of using the contextual conditions (Q4). In Figure xx.xx, it can be seen that a majority of users (91%, 21 people) gave positive ratings to this question and no negative ratings (value 1-2) were given.

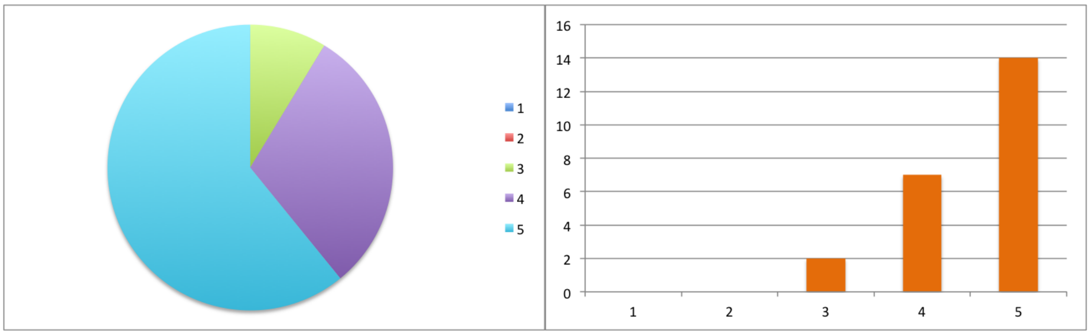

Figure 4.6: Distribution of ratings for understanding of using contextual information on a 5 point Likert scale (1, worst to 5, best).

4.4.6 Explanation Benefits

Explanations for critiquing exists in both variants of the system. For the context-aware recommender system built in this thesis, another explanation is added in the detail page for each item to explain why an item is recommended using the contextual information. In this evaluation, it is investigated in particular the satisfaction, usefulness and clearness of this newly added explanation for the context-aware variant.

Satisfaction

To test the general satisfaction of the contextual explanation, users were asked to rate how much were they satisfied with the provided contextual explanations (Q5). The average rating of satisfaction is 4.1. In Figure xx.xx, it can be seen that a majority of people (83%, 19 people) gave positive ratings (value 4-5) and no negative ratings (value 1-2) were given. However, among the positive ratings, more users (13 people) gave a more conservative rating (value 4) rather than the most positive rating (value 5).

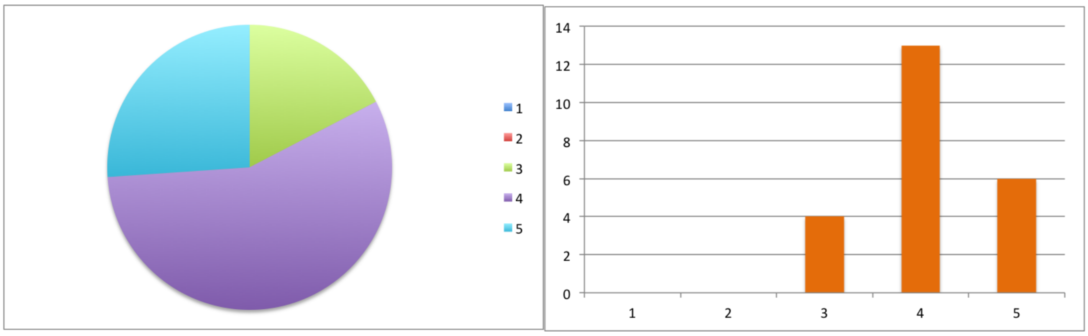

Figure 4.7: Explanation benefits in terms of distribution of ratings for satisfaction on a 5 point Likert scale (1, worst to 5, best).

Usefulness

To test the usefulness of the provided contextual explanations, users were asked to rate how much did they believe that the contextual explanations were useful (Q6). The average rating of usefulness is 4.3, which is larger than the average rating of satisfaction. In Figure xx.xx, it can be seen that a majority of people (83%, 19 people) gave positive ratings (value 4-5) and no negative ratings (value 1-2) were given. Among the positive ratings, more people gave the most positive rating (value 5).

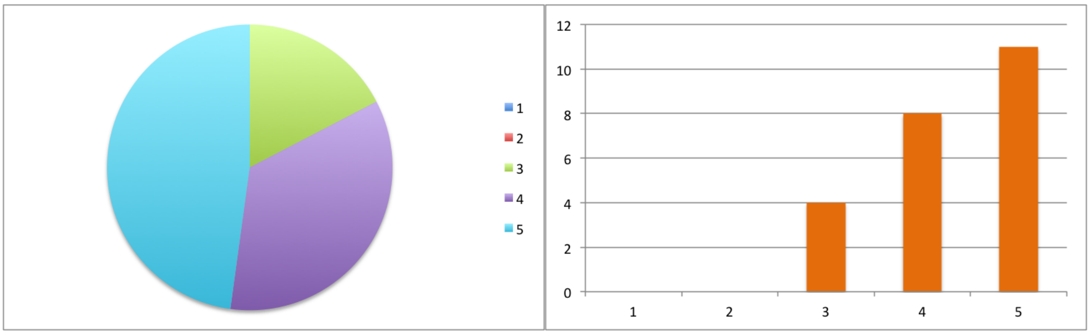

Figure 4.8: Explanation benefits in terms of distribution of ratings for usefulness on a 5 point Likert scale (1, worst to 5, best).

Clearness

To measure the clearness of the contextual explanation, users were asked to rate how clear are the contextual explanations provided by the system (Q7). The average rating of the clearness is 4.4. In Figure xx.xx, it can be seen that a majority of people (78%)

gave positive ratings (value 5). Among the positive ratings, 77% of the people gave the most positive ratings. Also no negative ratings (value 1-2) were given.

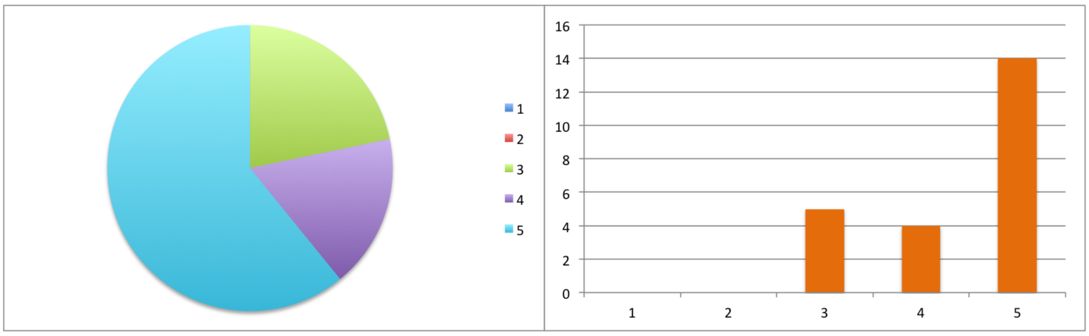

Figure 4.9: Explanation benefits in terms of distribution of ratings for clearness on a 5 point Likert scale (1, worst to 5, best).

4.4.7 System Preference

General Preference

To investigate the system preferences, users were asked to which system did they prefer more (Q8). In Figure xx.xx, it can be seen that 91% people (21 people) preferred CW to NCW. Among the two people who selected NCW, one of them still preferred the context-aware when they were asked which system provided more appropriate clothes.

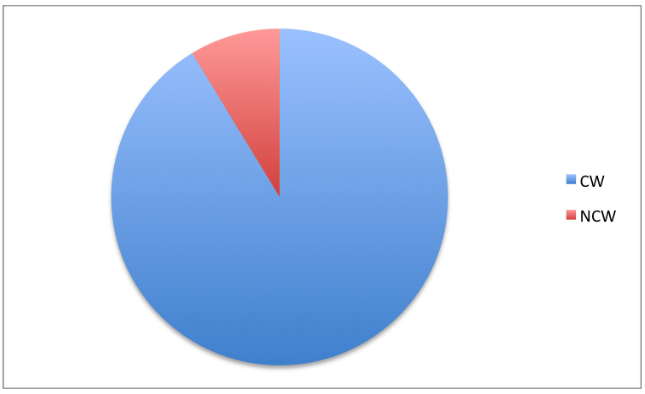

Figure 4.10: System preference in terms of distribution of general preference.

Satisfaction

Users were also asked to give a specific rating of how much did they like using these two systems to provide a more detailed view (Q3). In Table xx.xx, it can be seen that the mean rate of CW is higher than the one of NCW by about 0.65 and the difference is significant (with $p = 0.004 > 0.1$). Also the standard deviation of CW is smaller than NCW, which suggests that the users agreed more to the ratings of CW variant.

In Figure xx.xx, it can be seen that users shared a positive view towards CW and no negative ratings were given to CW, while two negative ratings were given to NCW. When looking at the distribution of the positive ratings, CW collected twice the number of very positive ratings (value 5) and twice the number of positive ratings (value 4) compared to NCW variant.

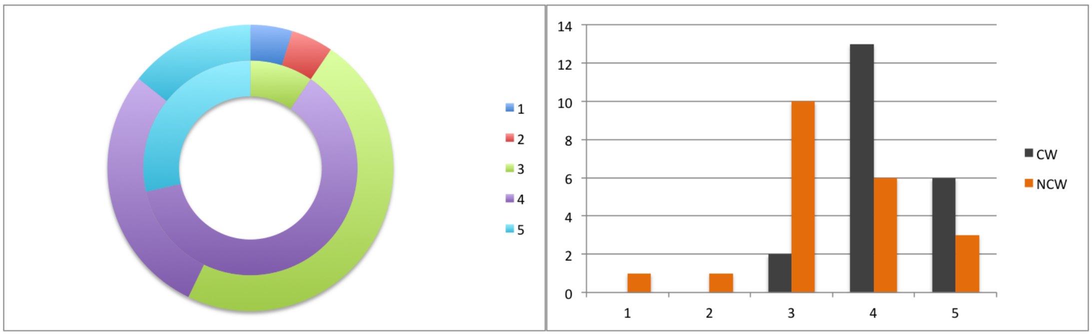

Figure 4.11: System preference in terms of distribution of ratings for system satisfaction on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right

4.4.8 Informal Feedback

During the test, users were encourage to give informal feedback, because it is users? most direct feelings of the system and can be used as guidance for further improvements of the system. We summarize the most relevant and most interest views here.

Allow manual input of the values for all available context factors. Two participants mentioned that they might not always want to buy clothes under the current context. It was also possible that they wanted to buy for future context. Thus the system should not automatically retrieve values for context factors such as weather, temperature, but let the user to enter the value by her/himself. The system can also be designed to allow the user to choose whether to let the system automatically detect the context value or let the user enter the value manually.

Improve explanation through using icons and color. From previous analysis, it can be seen that users value the usefulness of contextual explanations. There was a user who mentioned that instead of using only text, icons and different colors could also be used for displaying explanations. Images and colors are usually quicker than words and they can especially help to improve the user experience in mobile scenario.

Put location information in more conspicuous place. One user suggested that location, as a special context information, should be placed at a more conspicuous place (e.g., in the gridview where the user sees the list of all recommendations), because he thought that users were usually sensitive to location information.

Add a refresh button. Right now, the system only displays nice items for each cycle of recommendation. A few participants pointed out that a fresh button could be added. Instead of critiquing, they may want the system to show more items recommended based on the contextual information. It can be seen from here that the users valued the benefits of context and would like to rely more on it for recommendations.

4.4.9 Correlation Analysis

To understand the relationship between different measurements instead of only interpreting single measurement, a correlation analysis was performed on the measured data using the Pearson correlation coefficient. The correlation coefficient, denoted as r, measures the linear relationship between two variables and varies numerically between -1.00 and 1.00. The closer the value of r to 0, the weaker the relationship between the two variables is, with value 0 indicating an absence of relationship. The closer the value of r to 1 or -1, the stronger the relationship between the two variables is, with 1 indicating a perfect positive linear relationship and -1 indicating a perfect negative linear relationship. The values for all calculated r can be seen in Table xx.xx.

It can be seen that larger critiquing cycles were correlated with longer completion time (r=0.78). However, critiquing cycles and completion time are not correlated with the score of ease of use (r=0.04 for critiquing cycle, r=-0.11 for critiquing time), effectiveness (r=-0.08 for critiquing cycle, r=0.02 for critiquing time) and system satisfaction (r=-0.08 for critiquing cycle, r=-0.06 for critiquing time). Higher ratings for system effectiveness were correlated with higher satisfaction of the system (r=0.5).

When looking at the context benefits, it can be seen that the perceived benefits of using contextual conditions was not related to the general satisfaction of the current system (r=0.04). It was also not related to the clearness and the general satisfaction of the current system?s explanation (r=0.08 for explanation satisfaction, r=0.18 for explanation clearness). So it can be suggested that user?s preference of using contextual conditions is stable.

Table 4.2: The correlation analysis.

	EoU	E	SS	CB	ES	EU	EQ	CC	CT
Ease of Use	1	0.56	0.25	0.08	0.11	0.05	0.08	0.04	-0.11
Effectiveness		1	0.5	0.08	-0.1	0.03	0.36	-0.08	0.02
System Satisfaction			1	0.04	0.08	0.05	0.18	-0.08	-0.06
Context Benefits				1	0.3	0.48	0.27	-0.08	-0.27
Explanation Satisfaction					1	0.48	0.42	0.01	-0.03
Explanation Usefulness						1	0.59	0.1	0
Explanation Clearness							1	0.06	0.19
Critiquing Cycle								1	0.78
Completion Time									1

4.5 Discussion

Regarding prediction accuracy, the majority of people though that context-aware variant CW showed more appropriate than non context-aware variant. According to some informal feedbacks, clothes presented in CW looked better than those presented in NCW for the users. It might be because CW is recommending based on user's contextual information as initial preferences and can filter out clothes that are not satisfying. Also the clothes in the case base of CW were carefully selected manually according to expert knowledge, clothes with good design and cheaper cost can be discovered using collaborative effort which helps with boosting the recommendation accuracy.

Regarding ease of use, CW slightly beats NCW by 0.13, but the difference was not significant (p=0.3). However, users shared a positive view towards CW but gave 4 negative ratings (value 2) to NCW. It might be because users were give a way to express their contextual requirements and thus found the CW variant be more multifunctional and powerful. But it should also be noted that a too complicated user interface or unclear recommendation logic will also make the users be confused and rate the system low.

Regarding system effectiveness, CW significantly beats NCW by 0.35 (p=0.029). Since the users were asked to imagine a contextual scenario, users were more aware of the contextual requirement and thus value the context-aware recommendation more. There was one time, when the user was shown the context setting interface, the user commented that it was the exact thing he was looking for. On the other hand, more users gave positive ratings (value 4) than most positive ratings (value 5). It might be caused by their dissatisfaction of the missing of some other recommendation approaches that can be always seen in the online recommendation systems like Amazon. Some users commented that the system should allow the user to select the clothes category and gender first. Some other users would like to have a refresh button so that they can update the recommendations using the current context settings without critiquing.

Regarding context benefits, it is interesting to notice that the users all shared a positive view towards using context for recommendation and users gave twice the number of most positive ratings (value 5) compared to positive ratings (value 4). Thus it can be suggested that users are not against to using contextual information. In the correlation analysis, it was shown that user's preference of using contextual information was not related to the clearness and satisfaction of the current system's explanation. Thus it is not the special design of the system that highlighted the importance of context and it can be suggested that users are expecting recommender systems that can recommend based on their current context.

Regarding the contextual explanations, it was shown that users shared a positive view towards the provided contextual explanations provided by the system. The usefulness of the contextual explanation was scored high (average = 4.30), but the satisfaction of the explanation was scored a little bit lower (average = 4.09). It can be suggested that the importance of contextual explanations is valued high and users are quite sensitive to the quality of the explanations. The quality of the explanation can not be measured only by clearness, which can be indicated from the high score obtained by clearness (average = 4.39).

Finally, regarding system preference, more participants preferred using CW over NCW and significantly more participants were generally more satisfied with CW than NCW. Also a close relation was detected between system satisfaction and system effectiveness. It might be because for mobile application, users are more sensitive to the effectiveness of the system and the system effectiveness thus influences the general satisfaction of the system more than usual. Overall, the developed context-aware recommender system was proved to have successfully integrate contextual information into the conversation-based Active Learning system using case-based recommendation approach.

5 Conclusions and Future Work

In this thesis, a context-aware recommender system that integrate contextual information into existing Active Learning recommender system using case-based recommendation approach was developed and evaluated.

Following the methodology developed for developing context-aware, relevance of contextual factors was first acquired using a web tool developed to ask users to rate the influence of purchasing decision while imagining themselves being in selected contextual conditions. After the relevance was obtained, a case-based recommendation approach was proposed to integrate contextual information into the recommender system by recommending clothes items bought by other users under similar contextual conditions to the current user. A case base was set up as a knowledge base for the recommendation. To solve the static suggestion ability problem of case base, collaborative filtering approach was used for long-term knowledge engineering. However, no privacy issue need to be worried about because the correlation between users is performed at the session level and no user identification is required. To compute the similarity between cases, the Euclidean Overlap Metric (HEOM) was borrowed. With the learned factor importance before, contextual explanation for each recommended item can be easily generated by finding out the most influential factors for the current type of recommended clothes.

A mobile Android application Shopper using the developed recommender system was developed and evaluated and was shown to have a better evaluation regarding prediction accuracy, decision effort and general satisfaction compared to the baseline system that is not context-aware. Also users were shown to be well aware of the benefits of contextual information and expressed a strong demand for context-aware recommendation in the evaluation.

Several things can be done for future improvement. From the application design perspective, users can be allowed to control and specify values for as many context factors as possible. Right now, in the context setting view, values of part of the context factors such as weather or temperature are automatically detected if the users enabled them. However, during the evaluation, some users were complaining that the system was too intelligent by detecting the contextual information automatically and thus requirements for buying for future contextual scenarios can not be fulfilled.

For the proposed recommendation approach, right now clothes items are recommended as individual items to the user. In the future recommendation of clothes set can be realized through extending the current case structure by including more than one clothes item in each case. For example, a user who bought a coat in cold temperature may also buy a scarf that can match with the coat. Then this scarf can be included in the same case with the coat and be recommended as a set to another user.

The recommendation quality can also be improved by extending the case model by including the evaluation model. Right now, a case is stored in the case base if the user bought an item and the bought item is automatically rated evaluated as ideal item for the corresponding context. In the future, more approaches can be adopted for the evaluation of the idealness of an item. The item viewed by the user, the item with positively critiqued by the user, the stores most often visited by the user can also be used as hints for evaluation.

For the commercialization of the application, real data set instead of simulating one should be obtained from real offline stores, including the real time stock information for each clothes, the crowdedness of the store etc. However, this can be difficult because retailers are usually quite sensitive to the exposition of such data. This then comes to a business model problem. Furthermore, currently for evaluation purpose, the recommendation calculation is done on the mobile device and the database is also stored locally. In the future, they should all be migrated to a remote the server to reduce the burden of the client app and increase the system efficiency.

List of Figures

4.1 Prediction accuracy in terms of which variant provided more appropriate clothes. 39
4.2 Decision efficiency in terms of critiquing cycles. 40
4.3 Decision efficiency in terms of critiquing time. 40
4.4 Decision efficiency in terms of distribution of ratings for ease of use on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right 41
4.5 Decision efficiency in terms of distribution of ratings for effectiveness on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right 42
4.6 Distribution of ratings for understanding of using contextual information on a 5 point Likert scale (1, worst to 5, best). 43
4.7 Explanation benefits in terms of distribution of ratings for satisfaction on a 5 point Likert scale (1, worst to 5, best). 44
4.8 Explanation benefits in terms of distribution of ratings for usefulness on a 5 point Likert scale (1, worst to 5, best). 44
4.9 Explanation benefits in terms of distribution of ratings for clearness on a 5 point Likert scale (1, worst to 5, best). 45
4.10 System preference in terms of distribution of general preference. 45
4.11 System preference in terms of distribution of ratings for system satisfaction on a 5 point Likert scale (1, worst to 5, best). a) CW inner, NCW outer b) CW left, NCW right 46

List of Tables

4.1 Overview of the result. 38
4.2 The correlation analysis. 48

Bibliography

[1] Android design principles, July 2014.

[2] Android, the world's most popular mobile platform, July 2014.

[3] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.

[4] Robin Burke. Knowledge-based recommender systems. *Encyclopedia of library and information systems*, 69(Supplement 32):175–186, 2000.

[5] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. Developing constraint-based recommenders. In *Recommender systems handbook*, pages 187–215. Springer, 2011.

[6] Janet L Kolodner. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34, 1992.

[7] Fabiana Lorenzi and Francesco Ricci. Case-based recommender systems: a unifying view. In *Intelligent Techniques for Web Personalization*, pages 89–113. Springer, 2005.

[8] David McSherry. Similarity and compromise. In *Case-Based Reasoning Research and Development*, pages 291–305. Springer, 2003.

[9] Francesco Ricci, Bora Arslan, Nader Mirzadeh, and Adriano Venturini. Itr: a case-based travel advisory system. In *Advances in Case-Based Reasoning*, pages 613–627. Springer, 2002.

[10] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[11] Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias Höllerer. *I'm feeling loco: A location based context aware recommendation system*. Springer, 2012.

[12] S Shen and B Blau. Market trends: Mobile app stores, worldwide, 2012, 2012.