Context-Aware Recommendations for Mobile Shopping

Béatrice Lamche
TU München
Boltzmannstr. 3
85748 Garching, Germany
lamche@in.tum.de

Yannick Rödl
TU München
Boltzmannstr. 3
85748 Garching, Germany
yannick.roedl@tum.de

Claudius Hauptmann
TU München
Boltzmannstr. 3
85748 Garching, Germany
hauptmac@in.tum.de

Wolfgang Wörndl
TU München
Boltzmannstr. 3
85748 Garching, Germany
woerndl@in.tum.de

ABSTRACT

This paper presents a context-aware mobile shopping recommender system. A critique-based baseline recommender system is enhanced by the integration of context conditions like weather, time, temperature and the user's company. These context conditions are embedded into the recommendation algorithm via pre- and post-filtering. A nearest neighbor algorithm, using the concept of an average selection context, calculates how contextually relevant a recommendation is. Out of 20 clothing items from the hybrid recommendation algorithm, context-aware post-filtering searches for the nine best-fitting items. The resulting context-aware recommender system is evaluated in a user study with 100 test participants. The answers of the user study show, that the recommendations were perceived as being better than the recommendations of a non-context aware recommender system.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*Decision support*

General Terms

Design, Experimentation, Human Factors.

Keywords

context-awareness, mobile recommender systems, location-based services, user interaction, critiquing, mobile shopping

1. INTRODUCTION

Context-aware recommender systems (CARS) are systems utilizing the user's context such as the user's position, weather or social environment to recommend items. A context-aware recommender system could for example recommend the "Albertina" museum rather than visiting the "Prater" amusement park if the user spends a rainy day in Vienna. This paper evaluates which kind of context information is relevant in a mobile shopping recommender system and how this information could be utilized to improve recommendations of clothing items in a context-aware recommender system. By integrating contextual mobile information into the recommendations it is expected, that the recommended items better fit the customer's needs and therefore customers are more satisfied with the recommender system. The paper is organized as follows. We first start off with some definitions relevant for context-aware recommender systems and summarize related work. The next section defines the context factors and describes the system's overall design. The user study evaluating the developed system is discussed in section 4. The paper concludes by summarizing its results and giving an outlook on future research topics.

2. BACKGROUND AND RELATED WORK

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

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [6, p. 5].

They define context as relevant information for an interaction between a user and an application. Therefore, if the context of an entity shall be defined, it is necessary to ask which information is relevant to the situation.

Context-aware recommender systems (CARS) integrate context into the recommendation process. This process can be described by this three dimensional recommendation function [2]:

$$R\colon User \times Item \times Context \to Rating \qquad (1)$$

The rating function (R) considers the $Context$ (which is defined by all the different $Context\ Factors$) and recommends items of the item set ($Item$) to a user by predicting the

LocalRec'15, September 19, 2015, Vienna, Austria.

rating that this user would give to an item. Context complicates the recommendation process as items can be rated in different contexts. An umbrella for example can be rated at good weather conditions very highly, due to the fact that it looks nice or is small. However, if it was raining the same umbrella could get a bad rating, due to the fact that it breaks at the slightest wind. So the context in the rating function brings additional complexity as the recommendation algorithm does not only have to match users with items, but also with the context.

Adomavicius and Tuzhilin [2] identified three different points in the recommendation process where context might be incorporated into the process:

1. *Contextual Modeling* - the recommendation algorithm is altered such that it includes the context and already considers it when calculating recommendations

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

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

All of these approaches have their specific strengths and weaknesses. However, it is also possible to combine multiple context-based algorithms.

Since the consideration of context can enhance the usefulness of a recommendation for a user, CARS are recently receiving a lot of attention [1]. For instance Anand and Mobasher [3] define a recommendation process that integrates context. They distinguish between a user's *short term* (STM) and *long term* memories (LTM). Contextual cues are used to retrieve relevant preference models from LTM that belong to the same context as the current interaction. This information is merged with the current preference model stored in STM for generating context-aware recommendations [3]. However, the proposed framework is very general and does not emphasize how it can be applied in a mobile scenario, where the context is different.

Baltrunas et al. [4] investigated the relationship between contextual factors and item ratings in a tourist scenario. The authors developed a web tool for acquiring subjective ratings regarding points of interest in a mobile scenario within a specific context. Users were asked if a specific context factor (e.g. winter season) has a positive or negative influence on the rating of a particular item. Second, users were asked to rate example contexts and recommendations. The more influential a context factor seemed to be (according to the results of the first step), the more contextual conditions specifying this factor were generated. These imagined ratings could be used as initial ratings in the database, such that the cold start problem is minimized. Based on these results, a predictive model that can be trained offline, was developed. Results show that influencing context factors for points of interests are inter alia *distance, season, weather, time, mood and companion* [4]. This methodology seems to be a very promising approach to acquire contextual ratings, however ratings were only acquired for a travel planning recommender system and the generated ratings of this work can't be directly applied to a mobile shopping scenario.

Researches have also been done on automatically predicting the user's context. For example in [5], a mobile leisure recommender system was developed. It uses time, location, as well as personal data, such as calendar appointments, viewed documents and messages, to infer the user's current activity so that the user is not required to explicitly define her profile or preferences. The recommendations include stores, restaurants, parks and movies. However, up till now, the techniques for automatic context detection are often unreliable and immature and require further research [9]. We therefore decided to come up with a solution that takes the users' explicit stated preferences into account.

I'm feeling LoCo [10] is an ubiquitous mobile recommender system that recommends places nearby the user's current location, e.g. restaurants and museums. Physical context such as the user's current transportation mode and location are automatically detected. This physical information is used for a first filtering step: The user's mode of transportation and location influences the radius within which places for recommendations are considered. Moreover, the user's mood influences the recommendations: *foursquare* (a social network app to save and share visited places with friends[1]) assigns each place to a category, which is mapped by the authors to a particular feeling (e.g. the system recommends events related to Arts & Entertainment when the user feels "*artsy*"). As soon as the user states a mood, places assigned with the category to which the feeling is mapped to are recommended. The system is based on text classification. It considers the tags and categories associated with a place the user has visited. The user model is therefore a document, which holds all the names, categories and tags associated with a visited place. A conducted user study shows that *I'm feeling LoCo* enhances the user experience and that the recommended places were overall satisfying [10]. This mood-based approach is in particular reasonable if a recommender system is aimed to suggest different types of leisure activities since the user's mood might highly influence the current preferences. However, we consider the relevance of the user's mood as low in our mobile shopping scenario.

So far, no research exists that analyzed all the contextual factors that might be useful when recommending clothing items from different stores for mobile shoppers and investigated how such a recommender system can be constructed and is perceived. Such an application could help the user detecting new (formerly unknown) brands or stores and find clothes matching the user's fashion style. Compared to existing mobile recommender systems, clothing items are different in the way, that they frequently change. Such a recommender system has to be frequently trained or being able to provide good recommendations on a sparse dataset. We therefore first acquire the relevant context factors in a mobile shopping scenario and then come up with a promising approach how to integrate this context into the recommendation process.

3. DESIGNING THE PROTOTYPE

We imagine a system that uses the user's mobile context to recommend clothing items available in shops close to the user's position. However, as in our previously developed baseline system (see Section 3.1), the new approach should still allow critiquing of items. As described in Section 2, context can be integrated into the recommender system in three different ways: contextual pre-filtering, contextual post-filtering and contextual modeling. In this work,

[1]https://foursquare.com

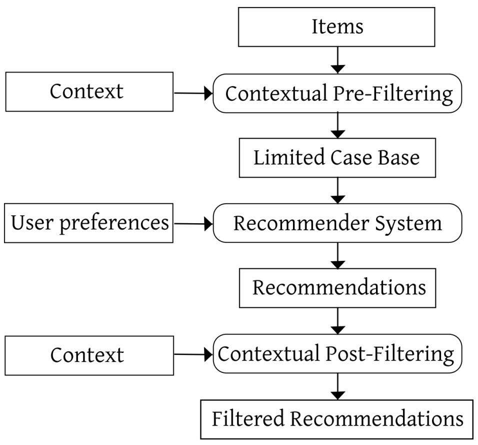
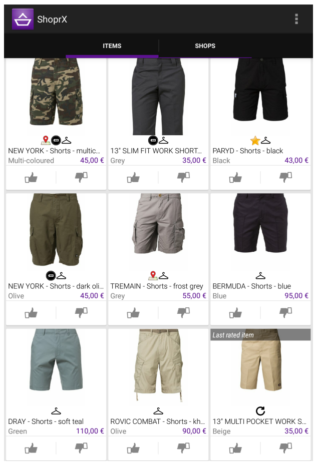
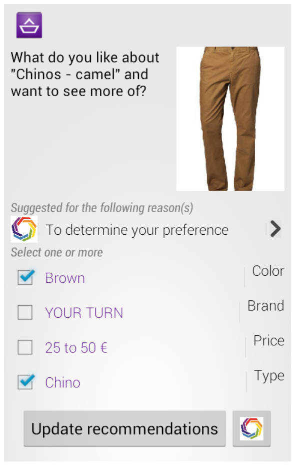


Figure 1: The context-aware shopping recommender process


ShoprX
ITEMS
SHOPS
NEW YORK - Shorts - multic... 45,00 €
13" SLIM FIT WORK SHORT... 35,00 €
PARYD - Shorts - black 43,00 €
NEW YORK - Shorts - dark oli... 45,00 €
TREMAIN - Shorts - frost grey 55,00 €
BERMUDA - Shorts - blue 95,00 €
Last rated item
DRAY - Shorts - soft teal 110,00 €
ROVIC COMBAT - Shorts - kh... 90,00 €
13" MULTI POCKET WORK S... 35,00 €

(a) Recommendations in CARS


What do you like about "Chinos - camel" and want to see more of?

Suggested for the following reason(s)

To determine your preference

Select one or more

✓ Brown	Color
YOUR TURN	Brand
25 to 50 €	Price
✓ Chino	Type

Update recommendations

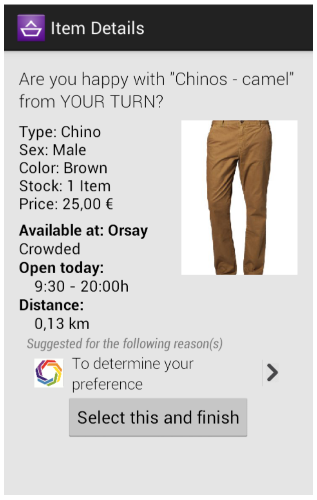
(b) Critiquing View in CARS

Figure 2: User Interaction Design

two approaches (contextual pre-filtering and contextual post-filtering) are combined to improve the recommendations (see Figure 1). Pre-filtering (Section 3.2) is used to determine which items of the case base are relevant to the user. Relevance for example depends on the distance the user accepts to travel, or the opening hours of a shop. Post-filtering (Section 3.4) is used to filter the items that shall be recommended according to their adequacy to the current context by using a nearest neighbor algorithm. In order to build a database of contextually tagged items, a pre-study was executed asking users to classify items according to contexts (Section 3.3). This data ensures, that some items already are contextually tagged, which is needed for the post-filtering of the recommendations. The user interface and interaction design of our CARS is described in section 3.5.

3.1 The Baseline

The system presented in [7] forms the baseline for our CARS. It was developed for the Android platform and incorporates an active learning algorithm. The user interfaces of the baseline system are very similar to our developed CARS and can be seen in Figures 2 and 3. The active learning algorithm, called adaptive selection, is a critique-based recom-


Item Details

Are you happy with "Chinos - camel" from YOUR TURN?

Type: Chino
Sex: Male
Color: Brown
Stock: 1 Item
Price: 25,00 €

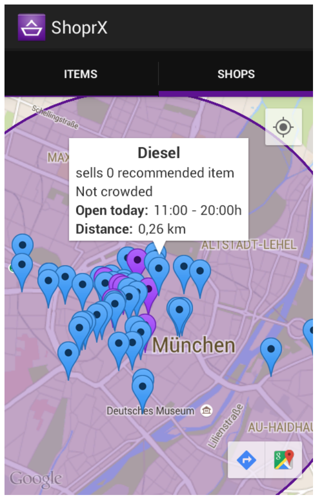
Available at: Orsay
Crowded
Open today:
9:30 - 20:00h
Distance:
0,13 km

Suggested for the following reason(s)

To determine your preference

Select this and finish

(a) Item view in CARS



(b) Map view in both systems

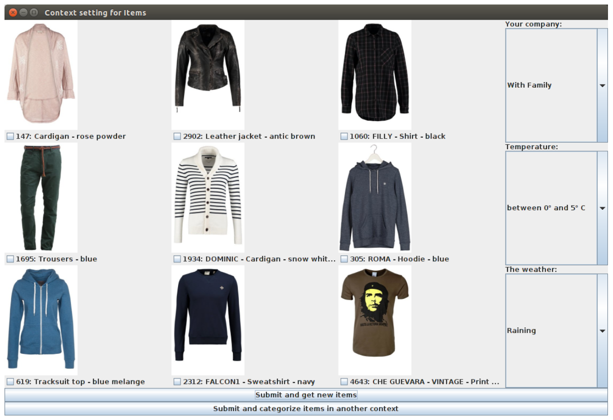
Figure 3: Detailed Information View

mendation algorithm. The algorithm uses a two-fold strategy: On a positive critique of an item (touching the thumbs up symbol) it shows items that more closely match the critiqued item. On a negative critique (thumbs down symbol) more diverse items are shown. In both cases, the recommendation algorithm uses a k-nearest neighbor algorithm to find the k items that best fit the current requirements. In this case k is set to nine, meaning that in each cycle, the user is shown nine different recommendations. In the following screen, the user selects which of the properties (color, brand, price, type) of the item shall be critiqued. The recommender system then shows more or less items (depending on the critique) of the selected feature(s). By touching an item's picture in the recommendations view, the system displays a result screen, where the user can select the item. The application also shows the immediate surroundings of the user in a map (see Figure 3). The system described in this section without context-awareness is used as a baseline for testing the context-aware recommender system. Furthermore we have made some adjustments to this content-based recommender system due to the changed dataset and performance problems.

3.2 Contextual Pre-Filtering

In the contextual pre-filtering step, we make sure that only relevant data is loaded into the recommender system. Therefore, the context factors *distance to shop, shop crowdedness, shop opening hours* and *item in stock* are used to restrict the case base and avoid unnecessary search in items the user does not want to see. The user may state preferences for each of these context factors. The user might state a different distance to the shops or that she wants to see crowded places as well.

The case base is filtered in four steps. First, all shops that are not within the specified distance, then shops that are not open at the specified time and shops that do not match the crowdedness criterion are excluded. Finally, it is verified that the item is in stock. After pre-filtering the items based on these conditions, it is verified that at least 300 items are available in the case base, as our tests showed


Context setting for Items
147: Cardigan - rose powder
2902: Leather jacket - antic brown
1060: FILLY - Shirt - black
1695: Trousers - blue
1934: DOMINIC - Cardigan - snow whit...
305: ROMA - Hoodie - blue
619: Tracksuit top - blue melange
2312: FALCON1 - Sweatshirt - navy
4643: CHE GUEVARA - VINTAGE - Print ...
Submit and get new items
Submit and categorize items in another context

Your company: With Family
Temperature: between 0° and 5° C
The weather: Raining

Figure 4: Tool for elicitation of item preferences in contexts

that this is the minimum amount of data to adequately react on the user's preferences. However, if there were not enough items available in the case base, these conditions are relaxed and the user is notified about this step.

3.3 Acquisition of Context Relevance

Before being able to recommend items based on context, the relevant context has to be defined. A promising approach to assess the context relevance for a tourism scenario is presented by Baltrunas et al. [4] and is therefore adapted for our shopping scenario (see also subsection 2). Using this methodology we assess the following context factors as relevant for our context-aware mobile shopping recommender system: *time of the day, day of the week, temperature, weather, company, distance to shop, crowdedness, shop opening hours* and *item is in stock*. In order to acquire contextual ratings, a convenience sample of the target population was asked to specify which items they are likely to buy in a specified context. We developed a simple Java tool (Figure 4) which shows nine pictures and descriptions of clothing items. The testers could specify if they would consider buying the product depending on a randomly selected company, temperature or weather, which is specified on the right side of the tool. Overall 747 contextual ratings for 674 different items were created by six users. This data forms the basis for the decision generation in the contextual post-filtering algorithm.

3.4 Contextual Post-Filtering

Based on the pre-filtered item set the critique-based recommender selects 20 items. Out of these 20 items only nine are actually displayed. Therefore, the contextual post-filtering algorithm (illustrated in algorithm 1) has to eliminate eleven items in each cycle. The context factors time of the day, day of the week, company, temperature and weather are used to post-filter the recommendations. For this purpose, we use a k-nearest neighbor method because this technique has proven to be adequate in different CARS. The most important component in nearest neighbor algorithms is the used distance metric. In our approach, the user is not able to rate an item within a given context, but only to select it (and therefore implicitly rating it as good). Based on this consideration, we came up with a distance metric that defines an *average context* in which an item is selected. The average context specifies in which context an item is selected. If an item was not selected in any context, it can be assumed, that this item neither is liked by a lot of users nor in a specific context and can therefore receive a higher distance to the current context. Popular items, which are selected in many different contexts will receive a distance which is close to 0.5. However, as they are very popular, they should not receive a high distance and therefore their distance is reduced by a defined percentage of their distance.

$$avgContextDist(c,i) = \frac{\sum_{b \in i_a} w_{i,b} \cdot dist(c_f, b)}{N(i_a)} \cdot \frac{N(f)}{\sum_{j \in i_j} w_j} \quad (2)$$

Equation 2 defines the distance metric. It calculates the distance between an item's (i) average context (in which the item is selected) and the current context (c). The first quotient calculates the average distance to the current context whereas the second quotient normalizes the distances.

The set of all context conditions in which an item has been chosen is defined by i_a. An individual context condition in which an item has been chosen is defined by b. For each clothing type, the context factors are of different importance. Hence, different weights ($w_{i,b}$) can be assigned to context conditions. We assigned the weights for each clothing type based on a previously conducted experiment [11]. The distance function $dist(c_f, b)$ (Equation 3) calculates the distance between the current context condition c_f (f stands for the context factor) and a context condition b in which the item was chosen. For an improved readability the variables were renamed to x and y in Equation 3. The number of context conditions in which an item has been chosen is defined by $N(i_a)$. In this work $N(i_a)$ always is a multiple of five - the number of context factors - as we assume all context conditions to be set in our artificial environment.

In order to make different items (with different overall weights) comparable, we normalize the distance between zero and one by multiplying with the second quotient of the function. Here $N(f)$ defines the number of context factors (five) we use for post-filtering. The number of context factors is divided by the sum of weights of all context factors (w_j) for the specific item ($j \in i_j$).

$$dist(x,y) = \begin{cases} graphDistance(x,y) & \text{if } y \text{ is nominal} \\ \frac{|x-y|}{range_y} & \text{otherwise} \end{cases} \quad (3)$$

If the context factor is ordinal, interval or ratio-scaled, the distances are calculated based on the euclidean distance. Otherwise the *graphDistance*, a pre-defined distance for nominal attributes, is used. This *graphDistance* is similar to the distance used by Lee and Lee [8]. The context factors weather and company use this *graphDistance* and define an undirected graph with distances between all context conditions (e.g. the weather conditions *Sunny* and *Rainy* have a higher distance than *Sunny* and *Cloudy*). The assigned distances are used as an input for the distance method. For the context factor time of the day we use a cycle, as the afternoon ends with the night, whereas the night is the first part of the day. For all other conditions it is expected that the euclidean distance provides good results. Although we want to achieve a high item frequency, we consider very popular items as being interesting for the user, especially in a shopping scenario. Therefore, we alter the resulting distance

($avgContextDist(c, i)$) if the item was selected in more than 30% of all contexts: The item's distance is reduced by 20 % so that it is more likely to be displayed to the user. Every item that is not selected in any context receives a distance of 0.51. We came up with this value because it is the average distance at the second tertile when considering all distances of items rated in a specific context to a randomly selected context. This ensures that items which have not been rated within a specific context in our pre-study (see Section 3.3) are more likely to be presented to the user than items that were considered as being uninteresting in that specific context. The whole algorithm for contextual post-filtering is presented as algorithm 1.

Algorithm 1 Post-filtering by current and item context

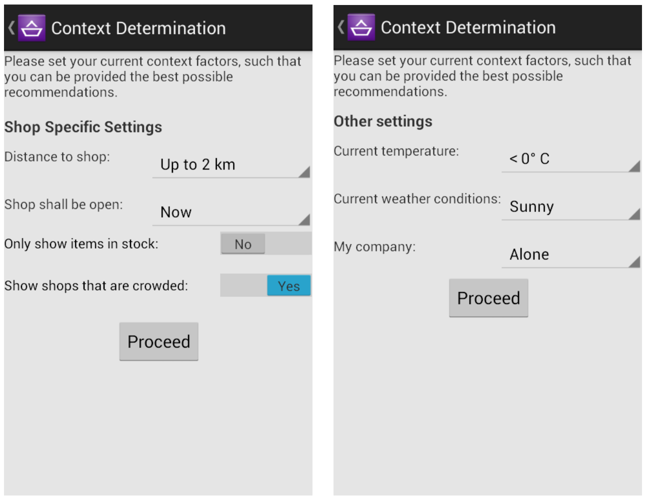
```
 1: procedure ContextPostFilter($items, context, k$)
 2:     for all $item$ in $items$ do
 3:         $avgContextDistance(context, item)$
 4:         if $itemDistance == null$ then
 5:             $setDefaultDistance(item)$
 6:         end if
 7:     end for
 8:     $decreaseDistanceForPopularItems(items)$
 9:     return $kNearestNeighbors(items, k)$
10: end procedure
```

The algorithm's disadvantage is that it weights each factor independently without taking into consideration possible connections between the individual context factors. For example the connection of rain and being with a friend might be more different from rain and being with the family, than the individual distances between being with the family and being with a friend. This detection of dependencies could be done by decision trees or other machine learning techniques. Nevertheless, we expect that the algorithm provides reasonable recommendations for the user's current context without these dependencies. The algorithm calculates the context distances in less than 100 ms on a Samsung Galaxy S3 mini for 20 items with the items being set in (overall) 200 different contexts. It allows weighting of context factors for each clothing type separately and distances for nominal attributes. The method $kNearestNeighbors(items, k)$ sorts the items by their distance to the current context. In case of any ties it uses the similarity measure that has already been applied in our baseline system (Section 3.1).

3.5 Navigation and Interface Design

When starting the application, the user is asked to set the following context conditions manually: preferred distance to the shop, opening hours, temperature, weather and company. Moreover the user can specify if she wants to exclude items that are not in stock and shops that are too crowded. The conditions for time of the day and day of the week, are not captured, as it is expected that the users are aware of these conditions subconsciously. The context determination interfaces can be seen in Figure 5.

Figure 2a shows an example of a calculated set of recommendations. With the *thumbs up* or *thumbs down* button the user is able to critique the item's attributes such as price, brand, clothing type and color (see Figure 2b). Besides this critiquing possibility, the user is able to see some explanations such as why the particular item is recommended.


Context Determination

Please set your current context factors, such that you can be provided the best possible recommendations.

Shop Specific Settings

Distance to shop: Up to 2 km

Shop shall be open: Now

Only show items in stock: No

Show shops that are crowded: Yes

Proceed

Context Determination

Please set your current context factors, such that you can be provided the best possible recommendations.

Other settings

Current temperature: < 0° C

Current weather conditions: Sunny

My company: Alone

Proceed

Figure 5: Explicit context determination via questionnaire

By clicking on an item's picture, the user gets to another screen with more detailed information about the item and the store. Here, the user can finally select the item (see Figure 3a). This information should enhance the trust the user has in the recommendations as she can check whether the initial preferences (about distances to shop, crowdedness, etc.) were incorporated. Moreover, we implemented a map showing all available shops. On click of a shop we show the shop's opening hours, the crowdedness, the name, the distance to the current position and how many items (out of the current recommendations) are available at this shop (see Figure 3b).

4. USER STUDY

The user study was designed in order to test the differences in user perceptions between the baseline application and the context-aware recommender system. We want to find out whether the users perceive a difference in the accuracy of recommendations. A second goal of the study is to find out whether users are more satisfied with a recommender system that takes the mobile context into account. Therefore, the goal of the user study is to evaluate if the following hypotheses are true:

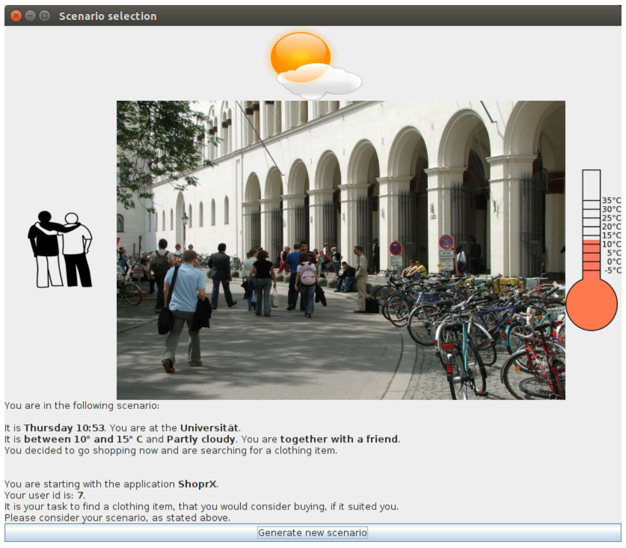
Hypothesis 1: The integration of context-awareness leads to better perceived accuracy compared to non-context-aware recommendations.

Hypothesis 2: The integration of context-awareness improves the overall user satisfaction.

Hypothesis 1 is tested by comparing the ratings of recommendations in a context-aware system and a baseline system. The users should rate how they perceived the recommendations on a seven-point Likert-scale. Hypothesis 2 shall test whether the users are more likely to use, reuse or recommend the application. This is an indication on how well the system adapts to the users and how satisfied they are.

4.1 Setup

The user study is designed as a supervised within-subjects user survey to minimize the number of survey participants



You are in the following scenario:

It is **Thursday 10:53**. You are at the **Universität**.
It is **between 10° and 15° C** and **Partly cloudy**. You are **together with a friend**.
You decided to go shopping now and are searching for a clothing item.

You are starting with the application **ShoprX**.
Your user id is: **7**.
It is your task to find a clothing item, that you would consider buying, if it suited you.
Please consider your scenario, as stated above.

Figure 6: Tool to generate a user's scenario

and improve the comparability between the applications. Each user tests both applications (the baseline system and the CARS) and answers a questionnaire afterwards. Which system is tested first is flipped in between subjects so that a bias because of learning effects could be reduced. The participants are asked to imagine being in the scenario, the tool generated for them, whereby the location is always Munich. The participant's task is to find one item only, which they would like to try on. As soon as the users have found a suitable item, they are asked to select it, such that they can finish the test and answer the corresponding questionnaire. The target population of this application are young smartphone users that like to go shopping. In the user survey qualitative and quantitative data are collected. Qualitative data is measured via a questionnaire. It mainly consists of statements, the user should assess on a seven-point Likert-scale (from 1 - strongly agree, to 7 - strongly disagree), e.g. how satisfied the user is with the recommendations and the application in general. The quantitative data is directly measured within the application and includes the number of critiquing cycles, the time between viewing the first recommendations and selecting an item, and the item diversity. Before the user starts using the application, a scenario describing the user's location, weather and company is generated for her (see Figure 6).

The participants are asked to actively select their context in the application and imagine it. This scenario is visually displayed to the users throughout the whole survey on a computer screen directly in front of them. The context conditions not mentioned in the scenario description, such as the crowdedness, can be selected by the user based on her own preferences.

The dataset used to test the application includes 5157 randomly selected fashion items, that were extracted from the Zalando API[2] of their UK-store in February and March 2015. Since our dataset is artificial, we distributed the items equally across all 129 shops and made realistic assumptions for our shops. The shop's opening hours were set to realistic values with moderate modifications to have some differences in the dataset. The crowdedness was set randomly with probability of 20 % and an item is in stock with a probability of 90 %.

[2]https://www.zalando.co.uk

4.2 Results

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

In order to test the user's perceived prediction accuracy, we asked if the recommended products fitted the individual preferences. The baseline application's mean is 2.71 whereas the CARS mean is 2.34 ($Median = 2$ for both systems). The Wilcoxon signed rank test reveals, that the recommendations of the CARS fitted significantly better to the user's preferences than the baseline's recommendations ($V = 1807$, $p < .01$).

The context-awareness of the applications is evaluated by asking whether the products were in line with the provided scenario. The baseline application's mean is 2.82 whereas the CARS mean is 2.66 ($Median = 2$ for both applications). The Wilcoxon signed rank test shows $V = 1346$, $p = .54$. This means that the users did not perceive any of the systems as being more context-aware than the other.

When asking the users whether they are likely to use the application again, the users stated that they are significantly more likely to use the CARS ($Median = 2$, $Mean = 2.64$) again, than the baseline ($Median = 3$, $Mean = 3.06$) application ($V = 1563$, $p < .01$).

The maximum time needed to find an item in the baseline application was 867 seconds ($Median = 142s$, $Mean = 179s$) and in the CARS application 697 seconds ($Median = 149s$, $Mean = 182s$). The time needed to select an item is not significantly different between the applications ($V = 2302.5$, $p = .45$).

Another measure for the effectiveness of the recommendation algorithm is the number of critiquing cycles until an item was selected. Participants completed their task in average 1.24 cycles less using CARS ($Median = 5$, $Mean = 6.1$ with CARS, $Median = 5$, $Mean = 7.34$ with the baseline system). Again a Wilcoxon signed rank test was executed ($V = 2393.5$, $p = .11$). However, the result is not significant, meaning that the null hypothesis cannot be rejected.

One of the goals of the CARS was to reduce the number of times an individual item is shown (*item frequency*) and thus increase the number of different items (*item coverage*). All in all the baseline application showed 7506 (1690 different; 22.5 % unique) and the CARS 6390 (1754 different; 27.4 % unique) items. We measured every time that an item was displayed to any user. The maximum number of times

Table 1: The means of some important measured values comparing both variations of the system.

	BASE mean	CARS mean	p value	V value
Perceived accuracy	2.71	2.34	**<.01**	1807
Perceived context-awareness	2.82	2.66	.54	1346
Intention to return	3.06	2.64	**<.01**	1563
Time	179 s	182 s	.45	2302.5
Cycles	7.34	6.1	.11	2393.5
Item frequency	4.39	3.62	**<.01**	285253.5

an item was shown was 115 ($Median = 3$, $Mean = 4.392$) for the baseline application and 53 ($Median = 2$, $Mean = 3.622$) for the CARS. A Wilcoxon signed rank test reveals that there is a significant difference between the samples ($V = 285253.5$, $p < .01$), meaning that the CARS showed items significantly less frequent than the baseline. Although the CARS showed less items overall, more different items have been shown. This indicates that the recommended items have been more diverse.

Overall, 59 participants reported that they prefer the context-aware application (CARS). This are significantly more compared to a random distribution of answers as a chi-squared test reveals ($\mathcal{X}^2 = 30.38$, with 2 df [degrees of freedom], p $< .001$).

The test participants found that the CARS recommendations fitted significantly better to their preferences. Therefore, hypothesis 1 that the recommendations by a context-aware system are perceived as better is retained. Hypothesis 2 that the overall user satisfaction is improved can also be retained to a certain degree as users were more satisfied with the CARS. The results might be less significant than expected as only six users rated items in context as an initial dataset. However, we wanted the dataset to be sparse as there are frequent changes to fashion collections.

5. CONCLUSION AND FUTURE WORK

In this work, a context-aware recommender system was developed and evaluated in a mobile shopping scenario. Our CARS is based on an active learning algorithm and uses a nearest neighbor algorithm. Compared to a system without context-awareness, the recommendations were perceived as significantly better in the CARS. Interestingly, the users did not attribute the better recommendation quality to the more context-aware recommendations but to better adaptability to their preferences and their clothing style, although the only difference from an algorithmic perspective is the context-awareness. It should be investigated in more detail, whether context-awareness is only perceived subconsciously. The next step for this application would be to test it in an online-experiment where real context-aware information is elicited. In a first approach the clothing data of some selected retailers would be enough to test this application online. In the future, we plan to conduct a user study where real context-aware information is elicited. Still a major challenge for context-aware applications is to acquire context-aware data to train or tweak a context-aware algorithm. For this user study, selected users classified the contexts in which they would try the clothes on. As the users in the user survey also imagined these contexts, we expect no significant differences between the classification of the items and the imagined scenario in the user study. This approach might help in narrowing down the problem of acquiring relevant context data as a quick start for a context-aware application. However, it has to be evaluated how close real contextual ratings can be estimated with this method. In order to adapt the existing approaches of estimating a rating to a yes or no decision we had to develop the concept of an *average context*, in which an item is selected. We believe that every context-aware recommender system relying on yes or no decisions might have benefits from adapting its context incorporation by using our approach. We also aim to find out whether the results of this work can be transferred to other application scenarios, such as for grocery shopping or leisure activity recommendation systems.

6. REFERENCES

[1] G. Adomavicius, L. Baltrunas, E. W. De Luca, T. Hussein, and A. Tuzhilin. 4th workshop on context-aware recommender systems (cars 2012). In *RecSys*, pages 349–350, 2012.

[2] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 217–253. Springer, 2011.

[3] S. S. Anand and B. Mobasher. Contextual recommendation. *Lecture Notes in Artificial Intelligence*, 4737:142–160, 2007.

[4] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context Relevance Assessment and Exploitation in Mobile Recommender Systems. *Personal Ubiquitous Comput.*, 16(5):507–526, June 2012.

[5] V. Bellotti and et al. Activity-Based Serendipitous Recommendations with the Magitti Mobile Leisure Guide. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, pages 1157–1166, 2008.

[6] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7, 2001.

[7] B. Lamche, U. Trottmann, and W. Wörndl. Active learning strategies for exploratory mobile recommender systems. *ACM International Conference Proceeding Series*, pages 10–17, 2014.

[8] J. S. Lee and J. C. Lee. Context awareness by case-based reasoning in a music recommendation system. *Lecture Notes in Computer Science*, 4836:45–58, 2007.

[9] F. Ricci. Mobile recommender systems. *Information Technology & Tourism*, 12(3):205–231, 2010.

[10] S. Savage, M. Baranski, N. E. Chavez, and T. Hollerer. I'm feeling loco: A location based context aware recommendation system. In *Advances in Location-Based Services: 8th International Symposium on Location-Based Services*, Lecture Notes in Geoinformation and Cartography. Springer, Vienna, Austria, 2011.

[11] W. Wörndl and B. Lamche. User interaction with context-aware recommender systems on smartphones. In *icom*, volume 14, pages 19–28, 2015.