Exploring the Design Space of Context-aware Recommender Systems that Suggest Mobile Applications

Matthias Böhmer, Gernot Bauer
Münster University of Applied Sciences
D-48565 Steinfurt, Germany
{mboehmer,gbauer}@fh-muenster.de

Antonio Krüger
DFKI GmbH
D-66123 Saarbrücken, Germany
krueger@dfki.de

ABSTRACT

Current technology development in mobile computing and upcoming application stores enable an easy development and distribution of mobile applications. This leads to an increasing number of available applications and to the user's problem of content discovery. Recommender systems aim at guiding users to relevant items. Currently, recommender systems that suggest mobile applications neglect that the usage of mobile devices is characterized by perpetual changes of a user's context. In this paper, we give rise to the context-aware recommendation of mobile applications. We explore the design space of recommender systems for mobile applications and describe the different dimensions and techniques for capturing the users, the items, the contexts and the corresponding relevances. For proof of concept we present the prototype of a recommender system that combines the design options in a hitherto unexplored way.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Current awareness systems; H.3.3 [**Information Search and Retrieval**]: Relevance feedback

General Terms

Design, Human Factors.

Keywords

Design space, recommender systems, mobile applications.

1. INTRODUCTION

While in former times mobile phones have been single purpose devices, nowadays smart phones serve as a generic platform for various kinds of services with different purposes, e.g. games, productivity, communication, travel guides, or navigation. The ubiquity, the hardware capabilities and the high-level programming interfaces of current smart phones have made them an appealing platform for application developers. Today, new applications can easily be distributed by developers and easily be installed by end-users. Since the number of available applications is steadily increasing the problem of content discovery [12] arises for mobile applications. Today, there are 106,000 applications available for the Android smart phone and 225,000 for the Apple iPhone [15].

Any of these applications has a special purpose and has been developed to support its users during a specific task or to meet certain needs. This can already be seen from the categorization of applications on application stores with classes like lifestyle, sports, and productivity. Once installed on a mobile device an application can be used anytime and anyplace. However, since mobile device usage is characterized by perpetual changes in the user's context [3], the set of required applications also changes.

Context can be defined as any piece of information that is relevant for a user's interaction with a system [7], e.g. on individuality, location, time, relations, and activity [19]. For our notion of context we refer to the user's activity. And since the mobile user's activity perpetually changes, so his need for different applications does. This context-dependency is neglected in current recommender systems that suggest mobile applications. For instance, it is not considered that most people will not play games while working, and that most users at home are more interested in applications for leisure time activities than in applications for productivity. Further, applications for scanning barcodes to retrieve product information can be used in different contexts, but in particular they make sense when the user has access to products with barcodes, e.g. during shopping.

Current systems that recommend mobile applications underachieve since they neglect the ever-changing context of their users and solely exploit sales figures and user ratings. Adopting Dey's definition of context-awareness [7] to the the domain of mobile applications a context-aware recommender system exploits context to provide relevant applications to its users, whereby relevance depends on the users' activities.

Mobile applications are a novel type of item to be recommended since (1) context plays a crucial role for the recommendation of mobile applications and since (2) the usage of a mobile application can be continuously captured alongside with the context information due to the nature of a mobile application – i.e. being a software process that can be observed. This goes beyond user ratings or implicit feedback based on sales statistics. In this paper, we explore the design space for recommender systems in the domain of recommending mobile applications.

CARS-2010, September 26, 2010, Barcelona, Spain.

2. RELATED WORK

The requirement of context-awareness within recommender systems has already been adopted in other domains of mobile computing. For instance, Belotti et al. [4] present a context-aware recommender system for leisure time activities on mobile devices and van Setten et al. [13] present a recommender system for tourists that shows points of interests. Yu et al. [18] investigate a context-aware recommender system for mobile multimedia content. The system of Yang et al. [17] provides location-aware recommendations on vendor offers according to the people's shopping needs.

However, hitherto there is only little research on recommender systems for mobile applications. The system for automatic menu customization presented by Fukazawa et al. [8] on the one hand presents content-based recommendations on the user's device, but on the other hand it does not consider the user's context. Woerndl et al. [16] present a recommender system for mobile applications that exploits context information and is based on capturing the installations of applications in relation to this context (basically location), though installation times are irrelevant compared to measuring the actual usage [10].

To the best of our knowledge, no comprehensive design space for context-aware recommender systems that suggest mobile applications has been described so far. This is what this paper aims at.

3. DESIGN SPACE

The main idea of context-aware recommender systems can be subsumized within a formula:

$$users \times items \times contexts \rightarrow relevance.$$

This mapping depicts that the relevance of an item depends on a user and his actual context, where relevance to our means describes the utility value of an application for a user in a certain context. It further spans the dimensions for the design space of recommender systems that need to incorporate context, as it is the case for recommending mobile applications. Nota bene, this value cannot be quantified a priori since it is determined by the user's cognition, but it needs to be formalized so that it becomes ascertainable for an algorithmic system.

3.1 Capturing the Parameters

To anticipate the relevance of a certain item for a user in a certain context the space of $users \times items \times contexts$ needs to be filled with values for the relevance. Based on these data as a knowledge base and by applying different algorithms, e.g. for collaborative filtering [2], those relevance values that have not been recorded beforehand can be estimated for the recommendation of the most relevant items.

In the following, we describe the design options for capturing these parameters in the domain of mobile application recommendation. For each parameter we will distinguish between an explicit and an implicit option. Explicit capturing means that the user is required to input some information and thereby gets interrupted from his current task. Implicit capturing means that the data can automatically be gathered from observations without any user disruption or effort.

The distinction between explicit versus implicit capturing is important and worth mentioning, since in the domain of mobile applications the technology gives new means to this parameter capturing. It is not only possible to track context information like the user's location due to the sensors that became common in mobile devices. The main and unique point for the recommendation of mobile applications is that the user's experience of an item takes place where the recommender system itself is running, i.e. on the user's device. Therefore, there are new technical opportunities to implicitly observe the experience and capture the relevance values. In contrast e.g. systems for book recommendations can hardly capture a user's experience of a book implicitly after he bought it, e.g. how long and how often he reads it. The user's experience of an item is an elementary parameter in recommender system since it impacts the relevance.

3.1.1 Ascertaining the Users and Items

Knowledge of the user's identity is fundamental for recommender systems. To give personalized recommendations the systems are required to relate all captured information to a user. A user himself could explicitly notify the system of his identity, e.g. by logging into the system or unlocking the mobile phone. For an implicit capturing of the user's identity it can be utilized that mobile phones are personal devices [6]. Therefore, all information that requires a reference to the user's identity can also be related to the user's device without any loss of information and without any efforts for the user. However, this implicit capturing is only possible when the recommender system is running on the device.

In the domain of mobile application recommendation the items are the applications. They can be either native or web-based. The system needs to be aware of the application that the captured parameters relate to. On the one hand, this can also be done explicitly by the user referencing a certain application or the system asking for the relevance of a certain application. And on the other hand, since the nature of the items is very close to the nature of the recommender system itself – i.e. running software – the items can also be tracked implicitly. On mobile devices the available applications can be queried, unless the mobile operating system is prohibiting access to required method calls.

3.1.2 Determining the Context of Use

While the concept of recording relevance values within a two-dimensional matrix of $users \times items$ is common within current recommender systems, the adoption of context has added a new dimension to recommender systems [1].

In order to give context-aware recommendations, the system needs to recognize the context of the user that corresponds to the point in time when the relevance of an application is determined. On the one hand pieces of information can either explicitly be announced by the user, or on the other hand implicitly be obtained from the sensors of the user's device as well as from other sources.

For explicit context capturing, the users are required to annotate their current context with attributes and provide information on their current activity. For instance, this can be done by providing keywords or choosing their current context within an ontology. Such ontologies for example can distinguish between *travelling*, *working* or *shopping*. It is worth mentioning, that this does not have to be done in real time. A user can also annotate his past activities or provide information on his future actions.

Current mobile technology also allows the implicit cap-

turing of information about the user's current context. For instance, location, time, and acceleration data can be used to reason on a user's activity [11]. Such data can be obtained in real time from the sensors of the device. Other sources for implicit information capturing are the calendar, conversations, and activity streams of social networks.

3.1.3 Quantifying the Relevance

The relevance parameter refers to the value that an item contains for a user in his current context. In the domain of mobile application recommendation this means the usefulness that an applications has for a user to solve his current task or to fulfill his current requirements, e.g. to recognize a song in a disco or to compare prices of products. As said before, this parameter cannot be captured or quantified directly.

An explicit capturing of the relevance requires the user's attention. However, some ways to capture the relevance of items within recommender systems have evolved and became common over time. Firstly, people can leave a free text comment for an application, which serves as a guideline for other users. On the one hand, free text comments are very expressive and can help other people to form an opinion on the relevance of an application, but on the other hand its semantic meaning is not directly readable by a machine. Further, absolute rating schemas like the 5 star Likert scale or the metaphor of *thumbs up / thumbs down* became widespread and are therefore easy to understand for users. They are also very clear in expression since the user directly transforms his experience into a computable value. Additional relevance feedback can be designed as a relative ranking of applications, e.g. by letting the user sort the applications.

Implicit relevance feedback on mobile devices is available through observing the user's device interaction. The relevance of an application therefore is correlated to certain measurable variables. Based on the assumption that an application is only relevant when it actually is used the application usage can be exploited for capturing relevance. For instance, it can be analyzed how often and how long an application has been executed. It can also be logged, that an application has been installed, uninstalled, or updated to a new version. Also, the arrangement of the icons of the applications is promising to be exploited as implicit relevance feedback [5].

3.2 Accessing the Recommendations

Besides the capturing of the presented parameters all the information needs to be aggregated, before recommendations on applications can be given to the mobile device users. The aggregation allows to query the data for the relevance of an application for a user in a certain context. In general, the idea of incorporating context information into recommender systems is not new, and algorithms can be adopted from existing approaches, e.g. for splitting the relevance measures of applications according to context [2].

The user's access to recommendations can also be designed in an explicit or implicit way. For an explicit recommendation access, the user himself will pull recommendations from the system. This is the case for the largest currently applied recommender systems for mobile applications, i.e. Apple's Genius on the iPhone and the Android Market Store on the Android phone [15]. Both recommender systems grant access to the recommendations directly on the user's device, when the user asks for recommendations. Implicit recommendation access should follow a push-based approach. Such an access can be designed directly via the mobile main menu, as presented by Vetek et al. [14]. The recommended applications can directly be started from the main menu, which in this case acts as a context-aware menu if there is no distinction between applications that are already installed and those that have not yet been used before.

4. DISCUSSION

The design space presented in this paper comprises four dimensions with two options each. It allows 16 different conceptual designs for systems in the domain of mobile application recommendation. Further, for each option different implementations are possible, e.g. for the explicit relevance feedback.

However, the combination of the design options might be constrained by the platform to be developed itself. For instance, there are currently web-based recommender systems for mobile applications, that cannot make use of an implicit parameter capturing. Further, not every configuration makes sense. For instance, an implicit capturing of the relevance values will be counterproductive due to additional user interaction, if an implicit capturing of the corresponding application is not possible. Though, the design options within each dimension are not mutually exclusive. Especially for the parameters of capturing context information and the relevance values explicit and implicit implementations can be combined. For instance, a user's rating on a 5 star Likert scale can be combined with the number of utilizations of an application.

The overall goal of a recommender system should be to support the user in the process of retrieving appropriate content. Therefore, a mobile recommender system should favor implicit over explicit parameter capture where possible to avoid additional effort for the user, especially because of the negative impact of task disruption [9].

With the inclusion of context as a new dimension the complexity of recommender systems increases. Firstly, new reasonable algorithms for data aggregation and relevance estimation are required to cope with this new dimension. Alternatively existing approaches can be adapted to make recommendations context-aware, like for instance context-based splitting of item ratings for collaborative filtering [2]. Secondly, the impact of the cold start problem increases. While hitherto the problem appeared if no data for a user or for an item was available, sparse data on the dimension of context will emphasize the problem. Thirdly, the amount of data within the system increases since it is multiplexed by the cardinality of the new dimension. Additionally the implicit parameter capturing in contrast to explicit capturing will allow to collect more meaningful data.

5. A PROTOTYPE IMPLEMENTATION

For proof of the proposed design space we built and implemented *appazaar*, a recommender system for mobile applications. It is realized based on the Google Android platform, since it provides APIs to access the required context information as well as capturing the application usage. Conceptually, the system consists of three parts: (1) a logger that runs on the mobile device for capturing the user's identity,

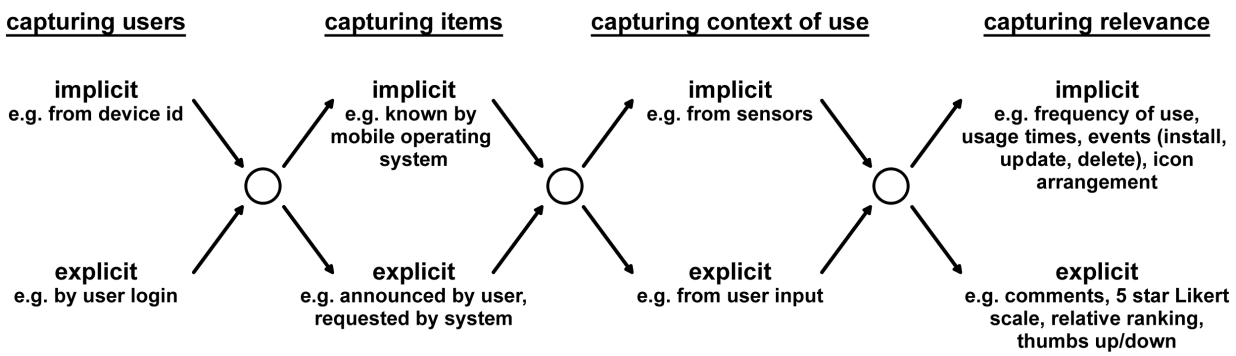

Figure 1: Design space for context-aware recommender systems that suggest mobile applications.

context information and the application usage; (2) a central unit that runs an inference engine to reason on the user's context and relates it to the application usage; and (3) a user interface that offers recommendations on applications to the users. The technical components are a mobile client and a server.

The mobile application implements the logger for the implicit parameter capturing and the recommendation interface. The logger is running as a background service and keeps track of the location including its accuracy, the local time and speed. Furthermore it records the start time of applications and their corresponding runtime. The identity of the user is equated with the technical identity of the device. The user is able to start and stop the logger and share his usage statistics, as figure 2a shows. All recorded data is stored in a local database on the mobile devices and is periodically uploaded to the server.

Figure 2b shows the user interface that visualizes the recommendations. It is realized as a widget that can be installed to the home screen of the device. It refreshes periodically, triggered by changes of context information (e.g. location), or when the device wakes up from standby. If the user clicks on an icon, the application either starts directly, if it already has been installed on the device before, or the click invokes an installation routine on the Android Market Store, if the application has not been installed before. As figure 2a shows, the user is able to turn off the recommendation of already installed applications. Currently we are not able to make this more transparent since we cannot automatically install applications due to security queries and possible costs.

The user is also able to insert keywords to describe his current context, as shown in figure 2c. We support two forms of input. On the one hand, a user is able to insert new tags by typing. And on the other hand the user is able to choose tags from a list of his recent tags or from a list of tags that other people used nearby the user's current location. The inserted context tags are used as explicit context information that allow the user to tailor the recommendation of the system. However, since the tags are also forwarded as a status message to a social networking site, those context tags become an implicit parameter when the user's primary motivation for entering them is related to the social network.

The mobile application uploads the recorded data to the server, which sends back the recommendations for applications. The server makes all incoming data persistent. It is used by the inference engine that applies clustering algorithms working on the sensor data to reason the contexts. Currently, the inference engine uses location and time as context information and builds contexts independent from a user's identity. Therefore a user that has never been in a specific context before receives recommendations on what other users have used in a similar context before. Additionally, the server maintains meta information about the applications (e.g. application type, icon, title) and about the devices (e.g. version of operating system).

6. CONCLUSION

In this paper, we described the need for context-aware mobile application recommendation and explored the design space for recommender systems within this domain. The novelty of investigating mobile applications as items for recommender systems is the ability to draw conclusions on the contextual relevance of an application from measuring the usage time and the usage frequency. We argued, that mobile application recommendation can benefit from integrating context into recommender systems, since it strongly influences a mobile user's needs. We explored the design space for context-aware recommender systems for mobile applications with a focus on parameter capturing and recommendation access. Different options for an implicit or explicit capturing of information about the user's identity, the applications, the contexts and the relevance values have been described. A brief discussion explained the advantage of an implicit in contrast to an explicit parameter capturing. We presented a prototype implementation of a recommender system for mobile applications called *appazaar*. This system recommends applications to mobile device users based on the actual usage of the applications as a relevance measure related to different contexts.

For future work we are going to investigate different algorithms for the aggregation of the ratings to our prototype and evaluate them based on real data. In addition we plan to evaluate different measures for implicit relevance feedback. It also seems promising to deduce to a user's context from the applications he actually uses.

Acknowledgment. This work was funded by the German Federal Ministry of Education and Research under grant 1748X08.

Figure 2: Screenshots of the prototype: a) settings to control the logging, b) access of recommendations within a widget, and c) input of contextual keywords.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Proc. of RecSys '08*, pages 335–336, New York, NY, USA, 2008.

[2] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proc. of RecSys '09*, pages 245–248, 2009.

[3] L. Barnard, J. Yi, J. Jacko, and A. Sears. Capturing the effects of context on human performance in mobile computing systems. *Personal and Ubiquitous Computing*, 11(2):81–96, February 2007.

[4] V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M. W. Newman, K. Partridge, B. Price, P. Rasmussen, M. Roberts, D. J. Schiano, and A. Walendowski. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *Proc. of CHI '08*, pages 1157–1166, 2008.

[5] M. Böhmer and G. Bauer. Exploiting the icon arrangement on mobile devices as information source for context-awareness. In *Proc. of MobileHCI '10*, 2010. to appear.

[6] A. Brush and K. Inkpen. Yours, mine and ours? sharing and use of technology in domestic environments. In *Proc. of UbiComp '07*, pages 109–126. 2007.

[7] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[8] Y. Fukazawa, M. Hara, M. Onogi, and H. Ueno. Automatic mobile menu customization based on user operation history. In *Proc. of MobileHCI '09*, 2009.

[9] A. K. Karlson, S. T. Iqbal, B. Meyers, G. Ramos, K. Lee, and J. C. Tang. Mobile taskflow in context: a screenshot study of smartphone usage. In *Proc. of CHI '10*, 2010.

[10] J. Nielsen. iphone apps need low starting hurdles. http://tiny.cc/eyie8, accessed on August 14, 2010.

[11] K. Partridge and B. Price. Enhancing mobile recommender systems with activity inference. In *Proc. of UMAP '09*, pages 307–318. 2009.

[12] B. Smyth, P. Cotter, and S. Oman. Enabling intelligent content discovery on the mobile internet. In *Proc. of IAAI'07*, pages 1744–1751, 2007.

[13] M. van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *Proc. AH 2004*, pages 235–244. 2004.

[14] A. Vetek, J. Flanagan, A. Colley, and T. Keränen. Smartactions: Context-aware mobile phone shortcuts. In *Proc. of INTERACT '09*, pages 796–799, 2009.

[15] Wikipedia. List of digital distribution platforms for mobile devices. http://tiny.cc/j0irz, accessed on August 14, 2010.

[16] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *Proc. of ICDEW '07*, pages 871–878, 2007.

[17] W. S. Yang, H. C. Cheng, and J. B. Dia. A location-aware recommender system for mobile shopping environments. *Expert Syst. Appl.*, 34(1):437–445, 2008.

[18] Z. Yu, X. Zhou, D. Zhang, C. Y. Chin, X. Wang, and J. Men. Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5(3):68–75, July 2006.

[19] A. Zimmermann, A. Lorenz, and R. Oppermann. An operational definition of context. In *Proc. of CONTEXT '07*, pages 558–571, 2007.