Utilizing Physical and Social Context to Improve Recommender Systems

Wolfgang Woerndl and Georg Groh
Technische Universitaet Muenchen
{woerndl,grohg}@in.tum.de

Abstract

Context has rarely been incorporated into recommender systems so far, but physical (e.g. a user's location) or social (e.g. the social network of a user) context can be useful sources for improving recommender systems. In this paper, we first discuss some principles for context-awareness in recommender systems. Then we present our hybrid recommender system for recommending applications to users of mobile devices. Finally, we describe our approach to utilize social networks to enhance collaborative filtering. Our evaluation shows that the social recommender outperforms traditional collaborative filtering algorithms in our scenario.

1. Introduction

While information retrieval approaches such as search engines like Google succeed in selecting suitable items (e.g. web links) according to a specific user query, these systems are usually not tailored towards particular user needs and preferences or the current context. That means, these systems return the same results for every user in every situation. Personalization and recommender systems adapt information access to a model of the user, but usually are not adapted to the user context so far [6]. A simple example is a restaurant guide running on a mobile device such as PDA. In the recommendation process, it is important to take the current context into account, because in a mobile scenario, nearby restaurants are preferable and a restaurant that is not open on the given day, should not be considered at all, even if it matches the user preferences very well.

The goal of the work is to identify general patterns for the integration of context into different types of recommender systems that utilize context from different sources. The context attributes we consider can be roughly grouped into "physical" context (such as the current user location, time and environment condition), and also "social" context (the social network of the user, buddy lists, past interactions etc.).

The rest of the paper is organized as follows. We first outline how to introduce context into the data model of recommender systems. In Section 3, we discuss the integration of context into the recommendation processes of different kinds of recommender algorithms in general terms. Then, we present our ideas for context-aware recommender systems in different scenarios, first a hybrid system that uses physical context such as the current user location to recommend mobile applications to users. In Section 5, we explain our approach to exploits social context to improve collaborative filtering in more detail. Finally, we conclude with a brief summary and outlook.

2. Recommender Systems Data Model

The main elements of the data model of recommender systems are users and items. Users are usually modelled by a set of attribute-value pairs (or, for simplicity, the respective vector of values only) which is also called user profile. The set of user profiles is denoted as U. In our view, it is beneficial to distinguish between profile and context information, which will be discussed below.

Items can have associated meta data, for example the location, price range and opening hours of a restaurant in a tourist guide, which is generally also modelled as a set of attribute value pairs (or vector of values respectively). The set of item-meta-data profiles is denoted as I. Thus items and users are technically modelled in the same way with respect to data structures.

Recommender systems traditionally operate on a user-item matrix. Technically speaking, this matrix is generated by a mapping $R: U \times I \rightarrow S$, mapping the values of the item meta data attribute-values and the user profile attribute-values to some appropriate mapping result-space S, e.g. a rating value in [0,1] describing this user's opinion about that item.

Our approach for the integration of context into recommender systems is to model context also as a vector of values of attributes, e.g. time, location or currently available network bandwidth in a mobile scenario. The space of context-profiles is denotes as C. We broaden the domain of the mapping R to U x I x C. In other words, context adds another dimension to the item-user matrix of CF (Figure 1).

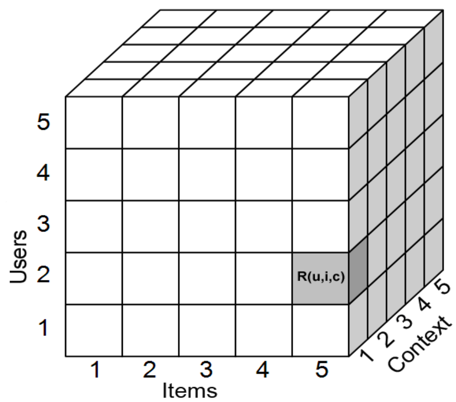

Figure 1. Integrating context in recommender systems: the user-item-context matrix

The introduction of context into recommender systems also adds an additional dimension of complexity (especially w.r.t. sparseness) to the recommender data model, because R may only be defined for a small part of possible contexts, e.g. ratings may be valid in one particular context only. Which context attributes are modelled, measured and used is largely dependent on the application domain.

3. Context in Recommendation Processes

In this section, we illustrate the integration of context into different types of recommender systems: content-based filter, collaborative filtering and hybrid recommender [5, 6]. After this discussion in general terms, we will present examples for context-aware recommender that take physical (Section 4) and also social context (Section 5) into account.

3.1. Content-based Filter

Content-based filtering recommends items that are similar to ones, the active user preferred in the past [3]. In general, to build a content-based recommender we need a filtering function F that uses the results of several mappings R_i (e.g. elements of the user-item matrix, or elements of a item-item matrix etc.). Typical content based filtering will for example rely on a user-item-rating R_1: U x I → S_1=[0,1] and on a matching between items: R_2: I x I → S_2=[0,1]. Matching between items can be done in different ways, for example by performing text analysis to find similar items or by identifying and applying rules. R_1 could, in principle, also be replaced by other indicators of (past) preference R': U x I → [0,1] (interest keywords from the user profile compared to item-meta-data etc.). The filtering function F: S_1^n x S_2 → [0, 1] would take the n best rated items and recommend a new item which matches these best, for example.

Context can be incorporated by using R'': U x I x C → [0,1] and thus being able to model "Adam likes Thai cuisine in the evening, but not in the morning". Furthermore we need (external) knowledge on the current context attributes for an user R_3: U x C → S_2={0,1}, e.g. modelling Adam's local time.

This abstract principle of matching users, items and context in a content-based filter has been implemented in our mobile applications recommender (see Section 4).

3.2. Collaborative Filter

Traditional collaborative filtering (CF) recommends items to an active that have been rated highly by users who are similar to the active user. Thus the filter function F exploits user-item-rating R_1: U x I → S_1=[0,1]. To model CF's filtering function F, two basic steps are needed to generate recommended items, neighbourhood creation and ratings prediction:

1. Selecting users that are similar to the active user. In most approaches, similar users are users that have rated users in a similar way to the active user in the past using. The user-user similarity usually is calculated by using Pearson correlation or likewise metrics [4].
2. Predicting the ratings of some items for the active user – based on ratings of neighbouring users – and displaying the best ranked items to the user.

Again switching to R'': U x I x C → [0,1] and using external knowledge about context R_3: U x C → S_2={0,1} allows for computing neighbourhoods that depend on context. It seems obvious that social context is important for neighbourhood creation among users, which we will discuss in more detail in Section 5 of this paper. As for the ratings prediction, the option is to apply a higher weighting to ratings that were made in a similar context in comparison to the current context of the active user.

The main drawback of CF is the new user and/or new item problem, because collaborative filtering needs a sufficient number of ratings, e.g. elements in the user-item matrix, to produce satisfying results. Therefore, CF is often combined with a content-based approach to form a hybrid recommender.

3.3. Hybrid Recommender

Hybrid recommender systems combine different approaches with the goal of utilizing the strengths of one algorithm while avoiding its weaknesses by applying a second approach. Combination strategies include weighted or cascading, switching or mixed, or feature combination or augmentation [5].

When integrating context, one option is to combine several (context-aware) algorithms to one hybrid context-aware recommender system. In principle, all of the above combination strategies could be used, but it is particularly promising to use a hybrid recommender to reduce the mentioned complexity of the user-item-context matrix (cf. Section 2). Thereby, a first algorithm RS 1 would operate on two dimensions and compute an intermediary result set of items. In a second step, the results are further filtered and ranked by taking the 3rd dimension into account. Thus, the used combination strategy is cascading or feature augmentation. Figure 2 illustrates this principle.

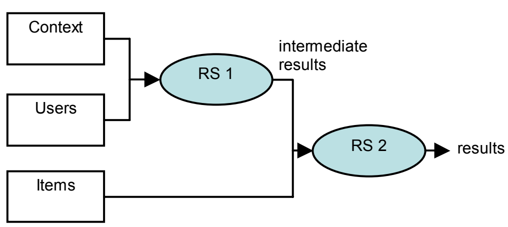

Figure 2. Hybrid recommender with context

3.4. Related Work

There are a few approaches that explicitly model and utilize context in recommender systems [3]. Several other systems incorporate context into user and item attributes, but we believe it is useful to consider context as a separate entity, as explained above.

Adomavicius and Tuzhilin also argue for a multidimensional approach to deal with context in recommender systems as an extension of the item-user model [3]. The paper introduces a multidimensional rating estimation method to deal with the additional dimensions. This approach generalizes to n dimensions (e.g. separate dimensions for time and location) in comparison to just 3 in our model, but we believe it is sufficient to model different context characteristics as attributes of one context vector. Item and user descriptions can also contain different characteristics such as price and weight of a product.

Chen utilizes physical context such as location to weigh ratings according to context similarity in collaborative filtering algorithms [2]. Thereby, a major problem is sparseness of available ratings in the same (or comparable) contexts.

4. Utilizing Physical Context

4.1. Developing Mobile Applications

This recommender system is integrated in a framework supporting the development of mobile applications. The framework provides standard components that can be reused for different mobile applications, for example modules to generate high level context from sensor data such as acceleration and location (GPS) sensors [7].

Part of the framework is a deployment server where developers of mobile applications can register their services and end users can browse and search for relevant and interesting gadgets. Users can access the deployment server and download client modules on their mobile devices. One problem for users is to find interesting and – with regard to their current context – relevant applications [1].

4.2. Hybrid Recommender System

The hybrid recommender system developed in this scenario recommends mobile applications to users derived from what other users have installed in a similar physical context (location, currently used type of device, etc.) [1]. Users can choose between several content-based and collaborative filtering components: LocationAppRecommender, CFAppRecommender and PoiAppRecommender.

The *LocationAppRecommender* selects mobile applications that were used in a similar context (most importantly, a nearby location) by other users in a first step. In a second step, the intermediary results are ranked according to ratings (positive/negative), i.e. taking other users into account. Therefore, this recommender is an implementation of the cascading hybrid recommender depicted in Figure 2.

The *CFAppRecommender* applies collaborative filtering in our domain of mobile applications using the *Taste* library [8]. Taste provides a set of components from which one can construct a customized recommender system from a selection of algorithms. The *CFAppRecommender* analyzes the item-user data without considering context in the first step and filters the intermediary results, i.e. recommended mobile applications, according to context attributes such as devices capabilities (display size and input capabilities) and currently available network bandwidth in the second step.

The *PoiAppRecommender* does not recommend point-of-interests (POIs) but recommends mobile applications based on POIs in the vicinity of the user (i.e. physical context) using triggers. An administrator can select among types of point-of-interests (such as restaurant, museum or train station) and specify within which radius of an actual POI an application is recommended. This is done when registering the application with the mentioned deployment server. When making a recommendation, the system then retrieves the current user position (using a GPS-enabled mobile device), determines POIs in the vicinity and generates a recommendation based on this context information. For example, an administrator can specify that her mobile train table application shall be recommended when the user is near a train station.

After applying the trigger rules, our approach uses collaborative filtering to rank found items according to user ratings of applications in a second step. User ratings are collected implicitly by automatically recording when a user installs an application within our framework. It is also optionally possible for users to explicitly rate applications after usage. The ratings are stored together with context information (time, location, used device, …) to capture the situation when a rating was made.

We have designed and implemented these recommender modules in the explained framework for the development of mobile applications. We are currently improving the components of our framework, developing more real world applications with student teams and then testing the system in practice by conducting a user study.

5. Utilizing Social Context

5.1. Social Recommender Systems

Besides physical context, social context is a key determinant for our actions and for our information needs. We thus propose to view models of social context as another important means to improve recommender systems.

There is good reason to assume that recommender systems in taste-related domains (music, movies, clothing etc.) should be different from recommender systems in rational domains (computers and technical systems, health-issues etc.). The ultimate quality criterion for recommender systems is the (subjective) usefulness of the recommendations, which has other factors of influence in case of taste-related domains than in case of rational domains. While in the latter case, objective utility measures can at least in principle be constructed, the former domains are more strongly influenced by social factors than by logic and mathematically weighing advantages and disadvantages. For example in social groups, group members and opinion leaders have great impact on the advice seeking and advice evaluation process of other group members in taste related domains.

The question is how social models can be used in a recommender system. The most obvious approach is to replace the neighbourhood creation based on ratings similarity in collaborative filtering (cf. chapter 3.2) by a "social neighbourhood" (e.g. all friends or all friends plus some or all friends of friends). Sinha and Swearingen compare conventional recommender systems' recommendations by the recommendations made by friends with respect to movies and books [10]. A similar investigation is done by Bonhard and Sasse [11]. Golbeck investigates a movie recommender system based on trust modelled by considering evaluations from friends in a stronger way [12].

We will now investigate how the traditional collaborative filtering style of neighbourhood creation performs in comparison to the social neighbourhood creation technique.

5.2. Empirical Studies and Evaluation

We conducted an empirical study to compare the two approaches in a taste-related domain [13]. The German community Lokalisten (www.lokalisten.de) is a Munich-based German language virtual community, which was founded in May 2005 and has (April 2007) approximately 700000 users all over Germany with an emphasis on the Munich area. The focus of the community is best described as communication- and spare-time-oriented. Central feature of the community is a simple social network model where two-way-handshake confirmed personal friendship relations can be created, managed and visualized by the users. Other than "friendship" no further types of relations exist. No weighting of relations is provided. From the relation graph a sub-graph of 4249 users was extracted. An online survey was constructed where those users where asked to rate 82 clubs in the Munich area on a discrete scale from 0 to 10 where 0 (which was pre-selected) indicates that a user does not know the club. 1012 users completed the survey successfully. Besides the 1012 x 82 rating matrix $M_{ur,}$ a similarity matrix S_{ij} was computed. The similarity between two users u_i and u_j is computed using simple cosine similarity on their rating vectors.

The first part of the experiment investigated whether rating congruency was at all statistically dependent on social relations, explained in detail in [14]. It could be concluded that groups of users with dense social relations have a significantly more similar rating behaviour than sets of people that do not have social relations [14].

In a second experiment the question was investigated how neighbourhoods computed on the basis of social relations compare to neighbourhoods computed via similarity of ratings in collaborative recommender systems. Some traditional collaborative filtering algorithms compute the neighbourhood with the help of thresholds: All users with a rating similarity above the threshold are included in the neighbourhood. Prediction computing is then accomplished in the simplest case by averaging the ratings from all users in the neighbourhood. In our data even for very low thresholds, the size of the neighbourhood is much larger than 4 and the number of larger cliques present in the friendship graph is very small. So cliques cannot be used as competing social neighbourhoods. For that reason, the friends of a user and all friends of these friends are taken as social neighbourhood for that user.

In order to account for effects of sparseness 25 increasingly sparse variants of the User-Rating matrix M_{ur} we computed by randomly setting ratings to zero. Of the 82 * 1024 = 82984 original entries, 25418 were > 0. The 25 variants of M_{ur} retain 25000, 24000, ..., 1000 entries. We compared the collaborative filtering neighbourhood creation with the social neighbourhood creation on each of the 25 variants of M_{ur}. For the CF approach thresholds 0.1, 0.2, ..., 0.9 for neighbourhood creation were tested nine and the threshold which performed best was taken und and compared to the social neighbourhood approach.

As quality measures that allowed for estimating the performance of the approaches, precision and recall, f-measure and mean absolute error were used. It is assumed that a prediction of a rating is adequate when it differs from the non-zero original rating by an absolute value of 1 (and 0.5 on the three most popular clubs of a user). The number of adequate predictions is denoted as P_a. The number of cases where the original rating was 0 and the predicted rating is non-zero is denoted as P_n (the number of novel predictions). The number of all other cases (in-adequate predictions) is denoted as P_i. The number of non-zero original ratings (25000, 24000, ..., 1000 in the 25 sparse matrices) is denoted as R_{pa}. The precision and recall and f-measure of every run are thus defined as

$$precision = \frac{P_a}{P_a + P_i} \qquad recall = \frac{P_a}{P_{ap}} \quad \text{and}$$

$$f\text{-}measure = \frac{2 * precision * recall}{precision + recall}$$

The mean absolute error MAE is defined as the arithmetic mean of the absolute differences between the predicted ratings and original ratings.

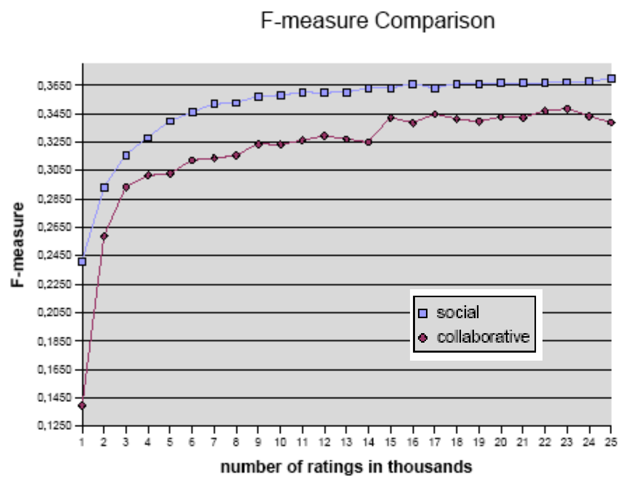

Figure 3: Comparison of F-measure performance of social recommender vs. CF-style recommender for various degrees of sparseness

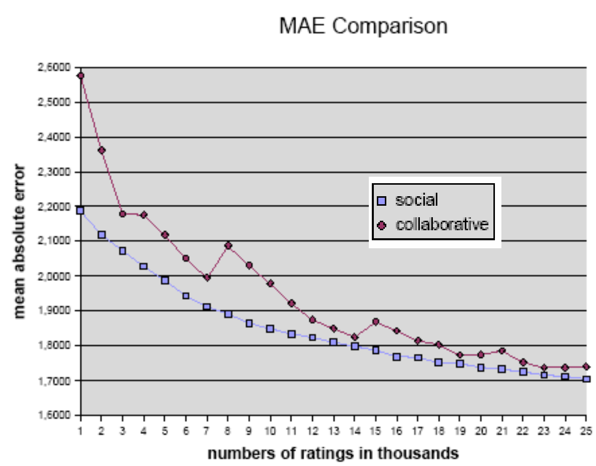

Figure 4: Comparison of MAE performance of social recommender vs. CF-style recommender for various degrees of sparseness

What we can see from Figures 3 and 4 is that the social neighbourhood based recommender performed better that the CF system. The jumps in the MAE curve for the CF system are due to changes in the neighbourhood-generation-threshold value λ. Since we optimized λ w.r.t. F-measure, changing from one λ to another may change the size of the neighbourhood which, in turn, influences the MAE-evaluation of the

CF system. The size of the neighbourhood is a critical factor for CF as discussed in [13].

Although it remains questionable whether these results can be generalized in all situations, it can be concluded that social neighbourhoods perform quite well in taste related domains. How can we use this type of context information to improve a recommender? First of all, trust and transparency are increased since recommendations come from known people. Secondly, the cold start problem of CF style neighbourhood creation can be avoided since the social neighbourhood does not depend on ratings. Thirdly, incorporating social neighbourhood has a positive effect on novel predictions. When novel predictions come from a user's peer group they are more likely to "broaden a user's horizon" in an accepted way due to normative effects of groups.

6. Conclusion

In this paper, we have identified and outlined some general ideas for utilizing context to improve recommender systems. We believe one benefit of this approach is that some principles can be applied for different recommenders in different domains. We have explained two applications scenarios that implement several of the possibilities, utilizing both physical context in our mobile application recommender, and also social context in collaborative filtering. Our social recommender can be seen as a substitute to a standard CF neighbourhood creation algorithm, but we are also working on combining the social network analysis with standard CF procedures. This can be done by using weights (which are based on the users' social networks) to compute user similarity (which is based on patters in the users' ratings). Other work in progress includes improving and then evaluating the mobile application recommender. We are also working on several additional application scenarios. One idea is to apply context-aware recommender systems in the domain of inter-networked cars [4].

7. References

[1] W. Woerndl, C. Schueller, and R. Wojtech, "A Hybrid Recommender System for Context-aware Recommendations of Mobile Applications", *IEEE 3rd International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces (WPRSIUI'07)*, Istanbul, Turkey, Apr. 2007

[2] A. Chen, "Context-Aware Collaborative Filtering System", Proc. of the International Workshop on Location- and Context-Awareness 2005, Springer LNCS 3479, Berlin/Heidelberg, 2005, pp. 244-253

[3] G. Adomavicius, and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 6, June 2005, pp. 734-749

[4] W. Woerndl, and R. Eigner, "Collaborative, Context-Aware Applications for Inter-Networked Cars". *Proc. 5th International Workshop on Distributed and Mobile Collaboration (DMC 2007)*, Paris, France, June 2007

[5] R. Burke, "Hybrid Recommender Systems: Survey and Experiments", *User Modeling and User-Adapted Interaction*, Vol. 12, Nr. 4, 2002, pp. 331-370

[6] S.S. Anand, and B. Mobasher, "Intelligent Techniques for Web Personalization", *Lecture Notes in Artificial Intelligence (LNAI) 3169*, Springer, Berlin/Heidelberg, 2005, pp. 1-36

[7] W. Woerndl, C. Schueller, and T. Rottach, "Generating High Level Context from Sensor Data for Mobile Applications", Proc. *IADIS International Conference Wireless Applications and Computing*, Lisbon, Portugal, 2007

[8] D. Lemire, and A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering", *Proc. SIAM Data Mining (SDM '05)*, 2005

[10] R. Sinha, and K. Swearingen. *Comparing Recommendations Made by Online Systems and Friends,* 2002

[11] P. Bonhard, and M. A. Sasse. "'Knowing me, Knowing you' – Using Profiles and Social Networking to Improve Recommender Systems", *BT Technology Journal*, Vol. 24, No. 3, 2006, pp. 84–98

[12] J. A. Golbeck, *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, College Park, MD, USA, 2005

[13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. *Evaluating Collaborative Filtering Recommender Systems*, 2004

[14] G. Groh, and C. Ehmig, "Improving Recommender Systems via Social Networks", *Proc. GROUP 07*, 2007, *to appear*