Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment

Annie Chen

IBM Zurich Research Laboratory,
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
ach@zurich.ibm.com

Abstract. In this paper we present a context-aware collaborative filtering system that predicts a user's preference in different context situations based on past experiences. We extend collaborative filtering techniques so that what other like-minded users have done in similar context can be used to predict a user's preference towards an activity in the current context. Such a system can help predict the user's behavior in different situations without the user actively defining it. For example, it could recommend activities customized for Bob for the given weather, location, and traveling companion(s), based on what other people like Bob have done in similar context.

1 Introduction

In the ubiquitous computing world, computing devices are part of the bigger environment, known as the *pervasive context*. These devices could be aware of various contexts in the environment, such as the location, the surroundings, people in the vicinity, or even the weather forecast. Connectivity, or the implied access to the information highway, is no longer bounded to the desk. The user could be at the train station, in a shopping mall or even in a different city. The role of computers in this environment has in a way become more like that of a personal assistant than, say, of a help-desk. This shift in interaction prompted us to look at how ubiquitous devices could assist users better by anticipating their preferences in a dynamic environment.

Currently, context-aware applications mostly rely on manually defined rules to determine application behavior for varying context. These rules can be predefined by application developers [1, 2] or, alternatively user-configured either by static preferences [3, 4, 5] or formed over time from user feedback [2]. Static rules are inflexible and difficult to customize for individuals, whereas the underlying learning process in the latter case has a long learning curve and can be tedious for users. More importantly, these systems are unable to predict a user's preference in an unseen situation.

It is understandably difficult for a computer to judge the taste of a user, so in recent years, we have seen a trend towards recommendation systems that leverage the opinions of other users to make predictions for the user.

Collaborative Filtering (CF) is a technology that has emerged in e-Commerce applications to produce personalized recommendations for users [6]. It is based on the assumption that people who like the same things are likely to feel similarly towards other things. This has turned out to be a very effective way of identifying new products for customers. CF works by combining the opinions of people who have expressed inclinations similar to yours in the past to make a prediction on what may be of interest to you now. One well-known example of a CF system is Amazon.com.

To date, CF has mostly been applied to web applications for which the context is undefined, i.e., the content is static, and the recommendations do not change with the environment. In the dynamic environment of ubiquitous computing, a user's decision can be influenced by many things in the surrounding context. For example, when people travel on holiday, their preferred activities might largely depend on the weather. Existing CF systems could not model this complexity of context. They are as likely to recommend mountain routes for a person who likes hiking whether it rains or shines. Applications in ubiquitous computing exist that have used CF to give recommendations [2, 7]; however they did not utilize the context information in the environment, and hence did not break out of the boundaries of a normal CF application.

In this paper we propose a design for a context-aware CF system in which we leverage the pervasive context information such that a user's preference is not only predicted from opinions of similar users, but also from feedback of other users in a context similar to that the user currently is in. In contrast to existing methods that manually determine how each context will influence the desirability of an activity, we use CF to automatically predict the impact of context for an activity by leveraging past user experiences. In our system, context provides the hints necessary to explore different options, rather than just limiting the set of options.

The remainder of this paper is organized as follows: we begin by examining the existing process in CF in Section 2. We then introduce context in Section 3.1 and define the requirements for a context-aware CF system. Next we discuss how to model context data in a CF system in Section 3.2 and how to measure similarity between different contexts in Section 3.3, and give an algorithmic extension to the CF process to incorporate context in Section 3.4. Finally we discuss future work in Section 4, and conclude in Section 5.

2 Background: Collaborative Filtering Process

The task of collaborative filtering is to predict how well a user will like an item given a set of feedback made by like-minded users [8]. An active user provides the CF system with a list of items, and the CF system returns a list of predicted ratings for those items.

Various classes of algorithms have been used to solve this problem, including Bayesian networks, singular value decomposition, and inductive rule learning [9]. The prevalent class of algorithm used is the *neighborhood-based methods*. In these methods, a subset of users is chosen based on their similarity to the active user, and subsequently a weighted aggregate of their ratings is used to generate predictions for the active user.

In this section we will describe the stages of neighborhood-based methods to provide a general understanding of a CF system and give the reader a point of reference for the latter sections, when we extend this process to incorporate context.

2.1 Building a User Profile

The first stage of a CF process is to build user profiles from feedback (generally in the form of ratings) on items made over time. A user profile comprises these numerical ratings assigned to individual items. More formally, each user u has at most one rating $r_{u,i}$ for each item i.

2.2 Measuring User Similarity

The key in CF is to locate other users with profiles similar to that of the active user, commonly referred to as "neighbors". This is done by calculating the "weight" of the active user against every other user with respect to the similarity in their ratings given to the same items.

The *Pearson correlation coefficient*, which measures the degree of a linear relationship between two variables, is commonly used to weight user similarity [10]. The similarity weight between the active user a and neighbor u as defined by the Person correlation coefficient is

$$w_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \bar{r}_a) \cdot (r_{u,i} - \bar{r}_u)}{\sigma_a \cdot \sigma_u} . \tag{1}$$

This equation combines the similarity between the relative ratings given by users a and u on the same item over all the items they have both rated. It gives a value between -1 and +1, where -1 means that these two users have the exact opposite taste, and +1 means they have the same taste.

2.3 Generating a Prediction

We can now combine all the neighbors' ratings into a prediction by computing a weighted average of the ratings, using the correlations as the weights [10]. The predicted rating of the active user a on item i can hence be formulated as

$$p_{a,i} = \bar{r}_a + k \sum_{u=1}^{n} (r_{u,i} - \bar{r}_u) \cdot w_{a,u} , \tag{2}$$

where n is the number of best neighbors chosen and k is a normalizing factor such that the absolute values of the weights sum to unity.

This covers the basic steps of the CF process in generating a prediction for a user. We now consider the issues involved when introducing pervasive context into the equation.

3 Incorporating Context into CF

3.1 Introducing Context: Concepts and Challenges

Context is a description of the situation and the environment a device or a user is in [11]. For each context a set of features is relevant. For each feature a range of values is determined by the context. From this model a hierarchy of context subspaces can be developed. Schmidt et al. [11] categorized context into six high-level subspaces. The first three relate to human factors: information about the user (e.g., habits, biophysiological conditions), social environment (e.g., social interaction, co-location with other users), and user's tasks (e.g., active tasks, general goals). The other three concern the physical environment: location, infrastructure (e.g., resources, communication), and physical conditions (e.g., noise, light, weather).

It is worthwhile noting that the first context identified on the list – knowledge about the habits of a user – is what a CF system currently models. CF uses this context to deduce any unknown habits of the user from habits of other similar users. What is lacking in CF is the knowledge about all the other contexts that help define a user's habits in different situations.

To model context in a CF system, we need to associate a user's choice or preferences with the context in which the user made that choice. This means that we need to capture the current context each time the user makes a choice. The same applies for the reciprocal: when a user asks for recommendations, we need to capture the current context and evaluate what others have chosen in a similar context in the past.

This poses two main problems: how do we manage context in the user profile in terms of data modeling and storage, and how do we measure similarities between contexts.

3.2 Context Modeling in CF

In a standard CF system, which we described earlier, a user's profile consists of a set of items with at most one rating assigned to each item. An item could be a product, a place or an action, and the rating represents the user's fondness of or preference towards that item. In a dynamic environment, a user's preference towards an item may change with the context. For example, Bob may want to visit a family diner instead of a posh restaurant when he is with his kids. To capture the different preferences towards an item in different contexts, a snapshot of the context need to be stored along with a user's rating for an item.

A snapshot of the context is a composite of different types of context data from various sources. This context can either be acquired from the embedded sensors in the mobile device itself or from an infrastructure placed in the smart environment which provides these data for the device. Consequently, various context data can be available or unavailable, depending on the infrastructure that is accessible in the current environment. This yields the requirement that different context types should be managed independently, and that their combined impact be calculated algorithmically.

In modeling context in CF, we took the approach of maintaining all the values and the structure within each context type. For example, the Location context object would maintain a hierarchy of all the different locations, such as "Czech Republic" and "Prague". Thereby we can minimize redundancy in the system and improve efficiency. When a user rates an item, the rating is associated with the current context value inside each available context, see Figure 1. In this example figure, Bob went to a spa when he was on holiday in Prague, he enjoyed it and rated this activity 5 out of 10. In the system a rating object is created that links Bob as the user and "Spa" as the item. There were two context data available to Bob's device at that time: the location and the temperature. To model this, the rating object also links to the value of "Prague" in the Location context and value of "5 degrees" in the Temperature context.

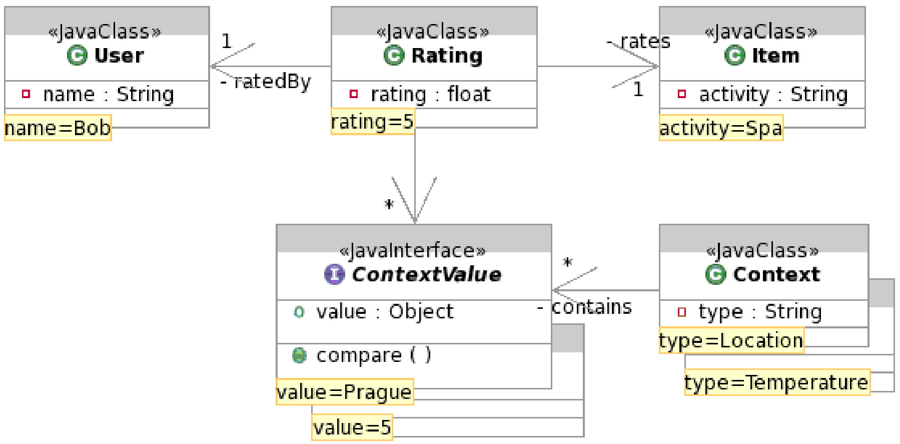

Fig. 1. Associating context with ratings

The links are bi-directional so that the system could easily traverse all the items rated for a given context and the entire context in which an item is rated. Context data is now available to the system, but to render it usable for making a prediction, we need to be able to compare the context of one user to that of another.

3.3 Context Similarity

The goal of calculating context similarity is to determine which ratings are more relevant for the current context. For instance, when Bob wants to go fishing in spring, ratings of fishing locations in spring would be more relevant than ratings of fishing locations in autumn. The similarity of the context in which an item is rated with the current context of the active user determines the relevance of this rating. Consequently, for each context type, there needs to be a *quantifiable measure* of the similarity between two context values.

Formally we define context here as a tuple of z different context types modeled in the system:

$$\mathsf{C} = (\mathsf{C}_1, \mathsf{C}_2, \cdots, \mathsf{C}_z), \tag{3}$$

where C_t $(t \in 1..z)$ is a context type (e.g., Location, Temperature or Time). For each context type t, there exists a similarity function $sim_t(\mathsf{x}, \mathsf{y})$, with $\mathsf{x}, \mathsf{y} \in \mathsf{C}$, which returns a normalized value denoting the similarity between x and y with respect to C_t.

A general comparator can be defined for a context type which has one of the following properties:

- *Categorical*, where values in the same category are alike (e.g., transport).
- *Continuous*, where closer values are more similar (e.g., temperature).
- *Hierarchical*, where a more general context can be used when no ratings for a specific context are available (e.g., location).

Context types which do not have these characteristics may require a custom comparator to be defined for them.

Context types can be vary widely, and it would be difficult to manually define a similarity function for each context type, so we devised an automated method to compare the relevance of one context value to another for the same context type.

We make the assumption that if user preferences towards an item do not differ much in different contexts, then the ratings given in one context would also apply for the other. So if the ratings for an item are similar for two different context values, then these two values are very relevant to each other.

We use the *Pearson's correlation coefficient*, which was used to calculate the similarity weight for a user in Equation (1) to measure the correlation between two different context variables with respect to their ratings. We denote the rating given by the user u on item i in context $\mathsf{x} \in \mathsf{C}$ as $r_{u,i,\mathsf{x}}$, and formulate the similarity weight for two different context variables, x and y, for item i as follows

$$rel_t(\mathsf{x}, \mathsf{y}, i) = \frac{\sum_{u=1}^{n} (r_{u,i,\mathsf{x}_t} - \bar{r}_i) \cdot (r_{u,i,\mathsf{y}_t} - \bar{r}_i)}{\sigma_{\mathsf{x}_t} \cdot \sigma_{\mathsf{y}_t}}, \tag{4}$$

where $rel_t(\mathsf{x}, \mathsf{y}, i)$ returns the *relevance* of two context values in C_t over all the ratings users gave in these contexts. Compare this with $sim_t(\mathsf{x}, \mathsf{y})$, which returns the *similarity* between two context values.

Either one or both could be used to weight each rating for the given context when making a prediction. The advantage of using the similarity function is that it can be better tailored to a particular context type, although it may be more restrictive in its comparison. Using the correlation function we were able to capture non-obvious relations between two context values, but each context value needs to be populated with many ratings for it to work.

Next we look at how this context is incorporated into the CF process to generate context-dependent predictions.

3.4 Context-Aware Extensions to the CF Process

Let us revisit the steps of a collaborative filtering process and redefine it to include context.

Building User Profile. In the context-aware CF system, a user's feedback needs to be put into context. This implies that the context needs to be recorded when the user selects or performs a recommendation. This poses a problem when the user wants to give explicit feedback on an item long after the action has passed and the context has changed.

Our advantage is that context data can also provide information on the user's current activity, which would enable the system to make implicit feedback for the user. Implicit feedback is when the system infers a user's rating for an item based on the user's behavior. Which behavior is monitored is application-dependent. For example, a tourist application could monitor how long a user stays in the location associated with a particular activity, whereas a shop application could look at items ordered, and a media-streaming application could see which items the user chooses to view or skip.

We could also use this implicit feedback to mitigate the problem of delayed feedback, by prompting the user to give feedback when the application senses the user has finished an activity (e.g. leaving a location or finished playing a clip). It could also allow the user to rate from a list of past activities the application has implicitly captured.

The feedback together with the available context would make up the user profile in the context-aware CF system as described in Section 3.2.

Measuring User and *Context* Similarity. In this step a normal CF system would "weigh" the active user against other users in the system with respect to similarity in their ratings to locate similar users. To incorporate context we would also "weigh" the current context of the active user against the context of each rating with respect to their relevance as described in Section 3.3 to locate ratings given in similar context.

Generating a Prediction. In this step a prediction is calculated by combining neighbors' ratings into a weighted average of the ratings, using the neighbors' correlations as the weights, see Equation (2).

In the context-aware CF system, each rating has an associated context. The similarity of the context in which an item was rated with the context of the active user determines how relevant this rating is in the current context, so we need to extend this to use the weighted rating with respect to the relevance of the rating's context to the current context.

We define $R_{u,i,\mathsf{c}}$ as the weighted ratings for the user u on an item i in context c, where c is the current context of the active user, using context similarity as weights. The context is multi-dimensional so we assume linear independence and calculate the similarity for each dimension separately, i.e.,

$$R_{u,i,\mathsf{c}} = k \sum_{\mathsf{x} \in \mathsf{C}} \sum_{t=1}^{z} r_{u,i,\mathsf{x}} \cdot sim_t(\mathsf{c}, \mathsf{x}), \tag{5}$$

where k is a normalizing factor such that the absolute values of the weights sum to unity. It has nested sums: the inner loops over each dimension in context, e.g., Location, Weather; the outer loops over all the values in that dimension, e.g., "Zurich", "Prague", "Tokyo" for the Location context.

We now substitute $R_{u,i,\mathsf{c}}$ for the rating of user u on item i without context $r_{u,i}$ in Equation (2). The predicted rating of the active user a on item i can hence be formulated as

$$p_{a,i,\mathsf{c}} = \bar{r}_a + k \sum_{u=1}^{n} (R_{u,i,\mathsf{c}} - \bar{r}_u) \cdot w_{a,u}. \tag{6}$$

This calculation combines all the weighted ratings, with respect to similarity in context, of all the neighbors, which is then further weighted with respect to the similarity of user, to give an overall prediction for the active user on an item in the current context.

4 Future Work

The next step is to evaluate the algorithms. The difficulty here is that we need real user data to validate whether the predictions match the user's actual decision. In addition, the collaborative nature of the system requires large user participation to generate good predictions. As we are the first to apply the CF technique to pervasive context, there is no readily available datasets to test it on. One possibility is to initialize the system with ratings from existing CF systems, such as VirtualTourist.com [12], adding any implicit context information available (e.g., location). This enables the system to provide general recommendations before more context data is collected to produce context-oriented recommendations.

We are currently developing a tourist application for mobile phones to demonstrate and collect data for the context-aware CF engine. We plan to test this application on a group of students in the laboratory, who will use the application for their weekend travels. The data collected from this deployment will allow us to evaluate and calibrate our algorithms.

On the modeling side, it would be interesting to look specifically into the social context and consider the influence of its complex interactions. Whether one should model relationships as a context (e.g., husband, girlfriend, children) or combine the profiles of individual participants.

Another important aspect to consider is the privacy issue. The system needs to keep usage statistics for each user in order to generate personalized recommendations. These statistics coupled with the context information can yield interpretations to potentially track everything from user movements to social behavior. Thus it is important that they be managed and protected properly.

5 Conclusion

We have designed a context-aware recommendation system that predicts a user's preference using past experiences of like-minded users. We used collaborative filtering to automatically predict the influence of a context on an activity. We defined the requirements of introducing context into CF and proposed a solution that addresses modeling context data in CF user profiles and measuring context similarity by applying CF techniques. Finally we gave an algorithmic extension to incorporate the impact context has on generating a prediction.

Much work still needs to be done to deploy the prediction engine in a real user environment, as we discussed in the section on future work. Predicting user behavior is an elusive art that requires iterative calibration to adapt to the user in a dynamic environment. This is why collaborative filtering could be very beneficial in solving this problem, because who could better to help us predict the users' preferences than the user themselves.

References

1. Xiao Hang Wang, Tao Gu, Da Qing Zhang, Hung Keng Pung: Ontology Based Context Modeling and Reasoning using OWL. In: Second IEEE Annual Conference on Pervasive Computing and Communications Workshops. (2004) 18–22
2. Mark van Setten, Stanislav Pokraev, Johan Koolwaaij: Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In: Adaptive Hypermedia 2004. Volume 3137 of LNCS. (2004) 235–244
3. L. Ardissono, A. Goy, G.P.: INTRIGUE: Personalized recommendation of tourist attractions for desktop and handset devices. Applied Artificial Intelligence **17** (2003) 687–714
4. Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, Christos Efstratiou: Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In: CHI. (2000) 17–24
5. Joseph F. McCarthy: Pocket RestaurantFinder: A Situated Recommender System for Groups. In: Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems. (2002)

6. J.B. Schafer, J. Konstan, J. Riedl: Recommender Systems in E-Commerce. In: Proceedings of the 1st ACM conference on Electronic commerce, New York, NY, USA, ACM Press (1999) 158–166
7. R.D. Lawrence, G.S. Almasi, V. Kotlyar, M.S. Viveros, S.S. Duri: Personalization of Supermarket Product Recommendations. Data Mining and Knowledge Discovery **5** (2001) 11–32
8. Jonathan L. Herlocker, Joseph A. Konstan, John Riedl: Explaining Collaborative Filtering Recommendations. In: Computer Supported Cooperative Work. (2000) 241–250
9. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, ACM Press (1999) 230–237
10. John S. Breese, David Heckerman, Carl Kadie: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. (1998) 43–52
11. Albrecht Schmidt, Michael Beigl, Hans-W. Gellersen: There is more to context than location. Computers and Graphics **23** (1999) 893–901
12. VirtualTourist.com: (http://www.virtualtourist.com)