Spatio-Temporal Proximity as a Basis for Collaborative Filtering in Mobile Environments

Alexandre de Spindler, Moira C. Norrie, Michael Grossniklaus and Beat Signer

Institute for Information Systems, ETH Zurich
8092 Zurich, Switzerland
{despindler,norrie,grossniklaus,signer}@inf.ethz.ch

Abstract. We propose a new approach to collaborative filtering in mobile tourist information systems based on spatio-temporal proximity in social contexts. Users store ratings and reviews of locations and events locally within their personal information system and then exchange these with other users present in the same social contexts using ad-hoc networking and opportunistic information sharing. We describe some preliminary investigations on the use of these methods in a mobile information system for visitors to a festival.

1 Introduction

Collaborative filtering (CF) techniques have become well-known through their use in on-line stores such as Amazon to recommend items to users based on buying patterns. The underlying assumption is that users who bought the same items in the past are likely to do so in the future. CF systems can either be *user-based* or *item-based*. In user-based CF, the system attempts to match users based on system-maintained profiles that may be a combination of explicitly supplied preferences and patterns derived from their actions. In contrast, item-based CF performs the filtering by matching items based on either similarity of features or simple association. For example, Amazon maintains a correlation matrix for items that indicates which items are related by the fact that a user purchased both of them [1].

Although there are many CF variants, what is common to current techniques is that they base their filtering on some form of stored profiles of users and/or items. These profiles may be based on information provided explicitly by the users or the producers of the items, but, since this is typically a tedious task, it is generally preferable to find some means of automatically generating profiles based on user actions. The profiles will typically be stored on a server and, on receipt of a request, a CF algorithm will be applied to perform the filtering and return the items most likely to be of interest to the user.

We were interested in providing CF in a mobile tourist information system designed to support visitors to an arts festival [2]. Our goal was that tourists should be able to carry their own personal copy of the festival information system with them on a portable computer, but somehow be recommended events,

venues, bars and restaurants based on ratings and reviews from other tourists. The system differs from traditional systems with CF in two ways. First, the system is designed so that it can deliver context-aware information to tourists without the need to have a network connection and therefore we did not want to have to introduce a server into the architecture. Second, unlike on-line stores, user and item profiles cannot evolve gradually over time. Visitors to the festival will want a filtering of relevant information and recommendations as soon as they arrive at the festival for what might be a visit of only one or two days.

We therefore decided to introduce a new variant of user-based CF that matches users based on *social contexts* rather than user interactions with the system. The idea is that users who go to the same place at the same time tend to have similar tastes. So if one festival visitor likes a particular style of trendy bar, then they may share other interests with the people who also like that bar such as theatrical events or contemporary restaurants. We therefore filter ratings and reviews based on *spatio-temporal proximity* of users as a basis for the formation of *social networks*. This works by users carrying their ratings and reviews with them and they are shared with other users in an opportunistic way based on ad-hoc network connections.

In Sect. 2, we start by motivating our approach using a scenario to explain the general operation of the system and the requirements that these place on both the system architecture and the form of CF techniques used. We then go on to discuss the CF methods we propose in detail in Sect. 3. Architecture and implementation issues are presented in Sect. 4. Concluding remarks and an outline of future work are given in Sect. 5.

2 Motivation

As tourists, we are continually faced with the problems of deciding where to go and what to see. Usually there is no shortage of information in the form of guidebooks, web sites and brochures. The problem rather is one of finding things that will suit our personal interests and tastes which is why we often turn to our friends or favourite series of guidebooks for advice. For visitors to large festivals such as the Edinburgh Fringe Festival with around 1800 events, the problems are even greater as, not only is the choice so large, but it is mostly unknown territory. While part of the fun is to take a chance choosing an event, many visitors will spend a lot of time reading the reviews in newspapers, pasted on the billboards and published on the Web. But with so many events, it takes time for critics to get round the events and it tends to be a relatively small set of visitors who enter their ratings and reviews on the Web.

Collaborative filtering systems have been developed to help users make a choice among unknown alternatives by trying to find other users with similar interests and tastes. The incorporation of other people's opinions in a decision process requires some form of social network where opinions can be selectively retrieved and combined. In the scope of this work, we simplify the main task of CF to inferring an opinion for a requesting user about a target item unknown

to them. We define an opinion as a rating value which is low to express a bad opinion and high otherwise. In order to generate recommendations and thus fulfil the claim of CF, we can think of ranking all items according to their ratings and selecting the highly ranked ones. In current research, CF approaches are frequently classified as being either *user-based* or *item-based*.

In *user-based* CF [3,4,5], the ratings from users similar to the requesting user are aggregated for the target item. The assumption is that similar users share similar opinions. User similarity is based on the comparison of user profiles representing a user in terms of features relevant in the scope of the application domain. In contrast, an *item-based* approach [6,7] first retrieves items similar to the one to be rated and then aggregates the ratings from all users about these items. In this case, the assumption is that similar items tend to be rated similarly. Another criteria makes a distinction between systems deriving the rating from analysing user behaviour [1] and others where users enter explicit ratings [8].

Sarwar et al. [6] and Aggarwal et al. [9] identify two main challenges that must be met by a CF system. On the one hand, the inferred rating must be accurate in the sense that it must be close to the rating made by the user if the target item were known to them. On the other hand, the inference process must be scalable. Neither the number of rating users nor the number of rated items must significantly slow down the process. Sarwar et al. also state that these two challenges are in conflict.

In both of the approaches mentioned above, one critical factor is how similarity between users and/or items is measured. The more similar the selected users or items are to the requesting user or target item, the more accurate the inferred rating will be. Improved similarity measures can be achieved with the incorporation of more features or by using more complex algorithms. In both cases, computation is slowed down. Consequently, simplifying the identification and selection of similar users or items can effectively speed up the inference process.

A related issue that arises in mobile information systems is the requirement for network connectivity. If all the information about users and items is to be stored on a server where the CF is performed, then this requires that each information request has to be processed by the server, requiring a network connection. This may increase costs to the users, while reducing both performance and availability of access. Further, in our example of a festival guide, many standard techniques for CF, which rely on constructing user profiles over long periods of system usage, may not be appropriate. A new visitor to the festival wants immediate recommendations about events and places. The question therefore is whether it is possible to find alternative means of inferring similarity between users that is both simple and immediate without requiring the user to explicitly construct a profile of themselves.

We propose a user-based CF technique which addresses the conflict between accuracy and scalability in mobile information systems. Users enter explicit ratings and exchange these. By using the spatio-temporal proximity of users in social contexts to determine user similarity, we are able to render the computation of

similarity of users unnecessary, while still generating accurate ratings. To illustrate the approach, we first present a simple scenario.

> *Mary is taking a break from attending events at the Edinburgh festival and is enjoying a late lunch in the restaurant of a film theatre. She enjoys the lively atmosphere with people coming in and out as festival films start and finish. She considers what event to attend next and consults her interactive festival guide. She decides on a play at a nearby theatre. As she is about to leave, she notices a man sitting in the corner studying the festival brochure who looks like her professor. After the play, she decides to go for a drink. There are a number of pubs in the same street, but her guide indicates a low rating, so she decides instead to go to one nearer her hotel which has a high rating. When she enters this pub, she immediately feels comfortable since it would seem that she is not the only solo festival visitor there. She sits at a table and enters her review of the play in her festival guide. Later on, she is surprised to discover the man who resembles her professor standing at the bar.*

Restaurants, bars and events define social contexts. What we do socially is based on our interests and tastes and therefore it is not surprising that we often see the same people in different places that we go to. Being in the same place at the same time, is an indication of social proximity. Mary has an interactive festival guide with a city map and information about events, restaurants and bars. While she is eating her lunch, her guide exchanges ratings with other guides in the same restaurant. When she consults her guide to see what events are on in the afternoon, the play she chooses is one with high ratings, many of which came from people present in the film theatre. At the same time, her guide also received a high rating of the pub near her hotel from the man who looks like her professor. It is not so surprising after all that she happened to see him there later.

Ratings and reviews are exchanged between users in an opportunistic manner that effectively performs a filtering based on social contexts. Users simply enter ratings and reviews into their guides, and the exchange of information is all performed transparently through ad-hoc peer-to-peer networking. Users are not aware of the exchanges taking place, nor do they know from whom they received the ratings. Two users will exchange ratings if they reside in the same place for longer than just a transient encounter. As detailed in the next section, this automatic exchange of ratings based on spatio-temporal proximity naturally leads to a user-based CF system.

Spatio-temporal proximity has also been used for filtering in the work of Xu et al. [10]. Their system comprises an extended notion of resources shared between users which are not restricted to ratings. However, the filtering is based on proximities between users and items described by the resources, rather than between users.

This work evolved out of a project to develop an interactive guide for visitors to the Edinburgh festivals based on interactive paper and an audio output channel. Users are provided with a printed festival brochure, a digital pen and an earpiece in addition to a portable computer coupled with a GPS unit. The

digital pen, based on Anoto technologies [11], tracks a user's interactions with the printed brochure via a camera embedded in the pen and a special position encoding pattern of tiny dots printed on the paper. An example of a brochure entry is shown in Fig. 1.

Fig. 1. Interactive festival brochure entry

A key feature of our system was to provide an easy means for users to input and access ratings and reviews. By touching pictograms on the paper with the pen, users can enter ratings of 1–5 for each event. The average of all ratings recorded can be retrieved by pointing to another pictogram. However, the system is not limited to numeric ratings only. Users can also submit their own comments on events by writing them in the blank comments pages provided at the back of the brochure and then linking them to an event in the brochure. This handwritten text is processed by Intelligent Character Recognition (ICR) software and can later be accessed by other users via a text-to-speech engine. Details of the system can be found in [12], including a description of the Anoto functionality and the general information infrastructure that we developed to support experimentation with mobile information systems.

In the previous version of the system, no collaborative filtering was performed and the ratings were simply averaged. Another disadvantage was that, while most of the functionality of the brochure could be handled by a mobile client without network access, the entry of and access to ratings and reviews required network access to a server. While we did consider the use of kiosks as special network access points for the uploading and downloading of ratings and comments, this together with existing CF techniques still did not meet our requirements for filtering based on social contexts. We therefore chose the option of an architecture based on ad-hoc peer-to-peer information sharing which essentially uses opportunistic network connectivity between peers to perform the user matching aspect of CF.

3 Collaborative Filtering

The most fundamental query in user-based CF consists of inferring a rating for a specific item by aggregating individual ratings from users with similar interests [9]. This process involves two stages, the computation of similarities that form the basis for the selection of the relevant users and the aggregation of their ratings about the specific item. In this section we show how users who share

ratings when they are in spatio-temporal proximity automatically perform a filtering of similar users. Finally, we also present an example aggregation method.

A rating is a tuple $(user, item, value)$ containing a rating user, a rated item and a rating value. Such a tuple can be seen as a directed weighted edge of a graph, pointing from a user node to an item node and weighted with the rating value. Therefore, a set of ratings defines a directed bipartite graph $G_r = (U \cup I, E_r)$ where U is the set of nodes representing users, I the set of item nodes and E_r the set of directed weighted edges.

The use of a graph to model ratings is not a novel approach. It has been proposed by Aggarwal et al. [9] and confirmed in subsequent works such as Mirza et al. [13] and Huang et al. [14]. We have chosen to represent ratings with a graph for several reasons. First, a graph captures all information based on which we infer ratings. Then, the use of a graph opens the rich variety of existing graph algorithms. Additionally, a graph representation allows network properties such as connectivity, clusters and cliques to be extracted and the construction of social network and recommender graphs [13].

Stage 1: Computing Similarity and Selecting User

Normally the first stage of user-based CF involves computing similarities between users and selecting the most similar ones. However, in our approach, the selection of users is performed implicitly and in the absence of any prior similarity computations. We now introduce some concepts forming the basis for our selection process of similar users.

User Similarity: The application domain of our CF system consists of users and items where users are visitors to the Edinburgh Festival using our interactive guide and items are things that they visit such as events, venues, bars and restaurants. If users consume an item, their location matches the location of the item for a specific period of time. Some items such as restaurants or bars can be consumed at any time within predefined opening hours and the duration of consumption can be anything from the time to drink a beer to eating a dinner. In contrast, items such as comedy shows or plays can be consumed only during a specific time period and the duration is usually well defined. We will refer to these two kinds of items as location and event items, respectively. Note that event items may happen only once or be repeated periodically.

Our selection of similar users takes advantage of the fact that, if two users consume the same item, they consequently find themselves at the same location. We cannot assume that users consuming the same location or periodic event items do so simultaneously, whereas, it is reasonable to assume so if they consume non-recurring event items. However, all items have in common the fact that, if users meet while consuming them, they probably stay in each other's vicinity for longer than if they would pass each other in the street. Users who consume the same items obviously share some similar properties such as interest and taste. If they meet when they consume location or periodic items, they also share

these properties simultaneously in absence of temporal constraints. This can be regarded as an additional similarity feature that they have in common. Based on these observations, we declare two users to be similar if they consume the same items simultaneously.

Spatio-Temporal Proximity: Such a similarity can be derived from a so-called *spatio-temporal proximity* of the users that covers two aspects of proximity, a spatial and a temporal one. If multiple users are consuming a particular item, their locations will match the location of the item. Thus, their spatial proximity will not exceed an item-specific boundary. If users are consuming an item simultaneously, the periods of time, during which they are consuming, overlap. This overlap is a result of temporal proximity. We conclude that if users are in spatio-temporal proximity, they are consuming the same items simultaneously and thus they are similar. Most importantly, the proximity implies that they have been in each other's vicinity at some time.

The history of item consumptions of a particular user u_a can be regarded as a set of *item consumption tuples* of the form $(item_i, [t_k, t_l])$ where each tuple is a composition of two entries. The first entry identifies a particular item $item_i$, and the second one defines the period of time $[t_k, t_l]$ during which the item was consumed. Thus, the history $H(u_a)$ of a user u_a can be written as

$$H(u_a) = \{(item_1, [t_1, t_2]), (item_2, [t_3, t_4]), \ldots\}$$

The condition for item consumption tuples to be equal is

$$(item_i, [t_k, t_l]) = (item_j, [t_m, t_n]) \iff (item_i = item_j) \wedge ([t_k, t_l] \cap_t [t_m, t_n] \geq p)$$

where we define $\cap_t$ as a temporal intersection of two time periods. The condition $[t_k, t_l] \cap_t [t_m, t_n] \geq p$ holds if the time periods overlap for a duration of at least p. The first component $(item_i = item_j)$ accounts for spatial proximity while the temporal intersection accounts for temporal proximity.

If two user histories contain equal consumption tuples, then they have consumed an item simultaneously. The parameter p determines the duration of temporal proximity required for consumption tuples to be equal. If the value of p is too small, users are assumed to have consumed the same item simultaneously as a result of a transient encounter. In contrast, if it is too large, users who actually do consume the same item simultaneously are not recognised as doing so. In the first version of our system, we have chosen a fixed value $p = 10$ minutes with the intention of ruling out transient encounters, while accounting for overlapping periods of consumption.

Spatio-Temporal Proximity as User Similarity Measure: The spatio-temporal proximity $P_{loc,t}$ between two users u_a and u_b can be expressed as

$$P^{\mathbb{N}}_{loc,t}(u_a, u_b) = \begin{cases} 1 & \text{if } H(u_a) \cap H(u_b) \neq \emptyset \\ 0 & \text{else} \end{cases}$$

This is a binary proximity measure in the sense that users are in spatio-temporal proximity if they have at least one tuple of their consumption history in common and they are not otherwise. It is reasonable to assume that the more often users consume the same item, the more similar they are. This calls for a second proximity measure that takes into account the number of common simultaneous item consumptions.

$$P^{\mathbb{R}}_{loc,t}(u_a, u_b) = \begin{cases} \frac{|H(u_a) \cap H(u_b)|}{|H(u_a)|} & \text{if } H(u_a) \neq \emptyset \\ 0 & \text{else} \end{cases}$$

On the one hand this continuous measure quantifies the degree of spatio-temporal proximity while, on the other hand, it also establishes a ranking of the users according to their proximity to the user denoted by the first argument. We say that two users u_a and u_b are similar if their binary spatio-temporal proximity $P^{\mathbb{N}}_{loc,t}(u_a, u_b)$ equals 1 or if it is greater than 0 for the continuous measure $P^{\mathbb{R}}_{loc,t}(u_a, u_b)$. A user u_a is more similar to a user u_b than to another user u_c if $P^{\mathbb{R}}_{loc,t}(u_a, u_b) > P^{\mathbb{R}}_{loc,t}(u_a, u_c)$.

Ad-Hoc Networking Entails Spatio-Temporal Proximity: Every user of our tourist information system maintains a local set of rating tuples. This set contains entries for item ratings made explicitly by the user as well as tuples received from other users. A user u_a receives rating tuples from another user u_b if they are within reachability for a duration of at least p. Thus, ratings are exchanged if the users consume an item simultaneously at least once. In terms of our proximity measure, this is the case if they have at least one item consumption tuple in common in their consumption history. Consequently, it also holds that

$$H(u_a) \cap H(u_b) \neq \emptyset$$

and therefore

$$P^{\mathbb{N}}_{loc,t}(u_a, u_b) = 1$$

With this and the fact that users only exchange ratings made by themselves, we conclude that if u_a exchanges ratings while consuming items, the local set of rating tuples contains tuples from similar users only where similarity is measured in terms of spatio-temporal proximity.

Stage 2: Aggregation of Item-specific Ratings

The second stage of user-based CF consists of aggregating rating values from similar users about the target item. As stated before, the deployment of a graph to manage the local set of rating tuples enables the use of arbitrary aggregation functions. The most common approach is to compute the average. In order to do so, we select all incoming edges of the node representing the target item and compute the average of their weights.

We can additionally take into account the degree of similarity as expressed with the continuous proximity measure. Therefore, we maintain a social network graph as proposed by Mirza et al. [13]. A social network graph $G_s = (U, E_s)$ contains the user nodes of a rating graph G_r. The edges in E_s are directed weighted edges pointing from a user u_a to a user u_b if $P^{\mathbb{N}}_{loc,t}(u_a, u_b) = 1$.This is the case between the local user and all users from which ratings are received. The weight of an edge pointing from a user u_a to another u_b is the value of $P^{\mathbb{R}}_{loc,t}(u_a, u_b)$.

If we are to infer a rating value from a requesting user u_r to a target item it_t we compute the average of the rating values contained in G_r, each weighted with the respective edge weights in G_s. When computing this weighted average, we only need the continuous proximity values for the rating user to all other users. The similarity between other users does not affect the aggregation and thus no continuous proximity information needs to be passed on when ratings are exchanged.

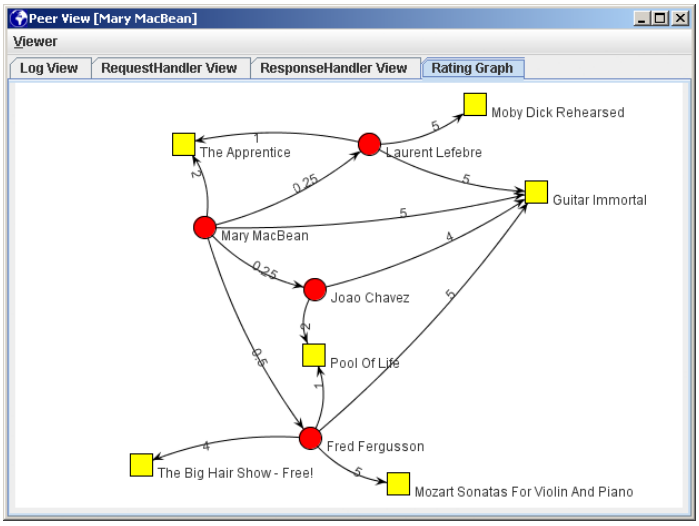

Fig. 2. Visualisation of rating and social graph

Figure 2 shows an example rating and social graph. The round vertices represent users and rectangular vertices visualise items. Edges between user vertices belong to the social graph whereas the ones pointing from a user to an item are part of the rating graph.

4 Implementation

The implementation of the CF system presented in the previous section requires components that discover users in the vicinity, exchange rating tuples and main-

tain a set of ratings made explicitly by the local user or previously received from remote users. A main component implements the aggregation procedure and serves as an interface to applications making use of CF such as our tourist information system. In developing an architecture and implementing these components, we were keen to obtain a flexible ad-hoc peer-to-peer service framework. The framework enables a straightforward integration with other applications and also establishes a platform for rapid prototyping of arbitrary peer-to-peer services. Multiple services can be deployed seamlessly and managed separately. We first describe the individual components and then show how they work together in order to implement our CF system.

Interaction Architecture

The general peer-to-peer interaction architecture shown in Fig. 3 forms the core of our framework. It provides the means for deploying peer services by specifying participating components and their roles in implementing the service interaction. In the figure, we reduce service interaction to the transfer of request and response data between a requesting local system shown on the left-hand side and a responding remote system on the right-hand side. One-to-many and many-to-one interactions can always be broken down to multiple one-to-one data transfers. In the local system, a request handler is used to post a local request. Depending on the concrete peer service, data may have to be provided by the requesting client. The handler creates a message object containing the request data and sends its XML string representation to a remote request handler ⓐ. On the remote side, the message is reconstructed from the string value received and the request data is extracted ⓑ. The remote request handler then processes the request which includes access to data structures and computational services yielding a response data object ⓒ. This response data object is wrapped with a message object and its XML string representation is sent back to the local response handler ⓓ. The response handler reconstructs the message and extracts the response data ⓔ. Typically, the response is then integrated into a local data structure.

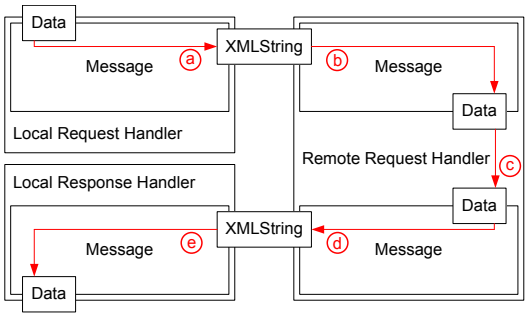

Fig. 3. Interaction architecture for peer services

We can identify three components each responsible for a particular aspect of service interaction: *handlers*, *data* and *message* objects. A `Data` class encapsulates the content transferred between peers while a `Message` class enables content-independent implementation of message formats such as plain XML, SOAP, binary, compressed or encrypted representations. Handlers implement concrete peer services in terms of request and response data processing.

As part of the framework, we provide abstract implementations of these classes covering the interface definitions expected by the components. A UML diagram of the `RequestHandler` and `ResponseHandler` classes declaring the methods to be implemented by a concrete service is shown in Fig. 4. Note that, unlike the architecture in Fig. 3, the local and remote request handler are put together into a single class.

This interaction architecture does not define a component responsible for building a physical channel through which messages are sent. Neither does it address the issues of discovering and identifying remote peers. We decided to separate the aspects of interaction and connection so that each can be implemented, extended and exchanged independently.

Connection Service

Our interaction architecture stipulates handlers processing local and remote requests as well as remote responses. These handlers form a logic communication channel between peers. They rely on a connection service providing the means to address a particular or multiple peers and to send and receive messages. This functionality is encapsulated in a connection service component. It consists of a peer abstraction that can be identified by other peers, a connection handler creating and maintaining physical channels to remote peers and a networking layer discovering and identifying remote peers. We have previously implemented such a component based on existing technologies including the JXTA [15] peer-to-peer framework and Apache Axis [16] web services. Due to the lack of existing ad-hoc networking frameworks, we implemented our own ad-hoc networking layer. The left part of Fig. 4 shows the UML diagrams of the `Peer`, `ConnectionHandler` and `AdHocNetworkingLayer` classes that provide the ad-hoc connection service to the request and response handlers.

The connection handler defines an interface method for sending messages to remote peers. In our case, the request message is always sent to a particular peer which is why a peer identifier must be provided. A request handler features a reference to a connection handler in order to post local requests. The connection handler has a reference to the local remote request handler, i.e. the handler notified upon incoming remote requests. The latter returns the response message which is forwarded to the requesting peer. It also has a reference to the remote response handler which is notified when a remote response arrives.

The peer abstraction is responsible for starting and stopping a connection handler and it maintains a collection of running connection handlers. Each connection handler contains a pair of service-specific request and response handlers. The respective request and response handlers must be registered with the

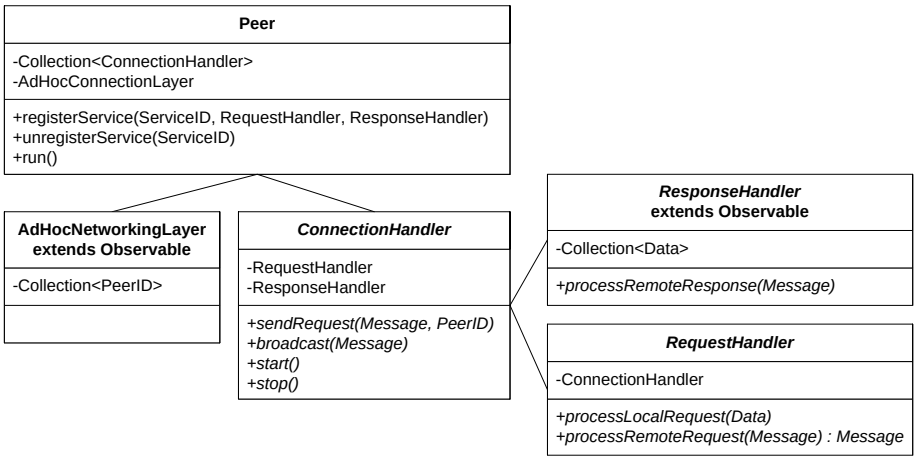

Fig. 4. Ad-hoc peer-to-peer platform

peer to deploy a particular service. The peer creates a new connection handler object, assigns the handlers to it and sets the reference in the request handler. We can deploy multiple services using the same class of connection handlers, in which case, each of them forms a logical peer-to-peer channel while they all use the same physical connection. On the other hand, we can have multiple connection handler classes, each implemented for a different connection technology. In this case, technologies such as JXTA, web services and ad-hoc networking can be used in parallel.

The peer is also responsible for creating and maintaining the networking layer. This ad-hoc networking layer encapsulates the tasks of discovering and identifying peers entering the area of reachability. It features an event mechanism notifying registered listeners about the arrival or departure of reachable peers. It can be extended to make use of any connection technology such as Wireless or Bluetooth. In the scope of this work, we have developed an extension for IEEE 802.xx wireless facilities.

Collaborative Filtering

We have implemented two classes, the `RatingModel` and `RatingManager`, shown at the bottom of Fig. 5, to manage rating tuples and perform rating inference. The rating model maintains a graph containing all ratings received from other users or made explicitly by the local user. For our implementation, we used the Java Universal Network/Graph Framework (JUNG) [17] since it contains basic graph algorithms and features visualisation facilities useful for debugging. The rating model features methods to add and retrieve rating tuples as required by the remote request and local response handlers. The main class of our CF system, a rating manager, defines an interface to other applications. One method allows the local user to make an explicit rating which is treated equally to ratings from

other users. It is added to the graph model and hence is included in the aggregation rather than replacing it. Another method takes a target item identifier as argument and returns the inferred rating as presented earlier in Sect. 3.

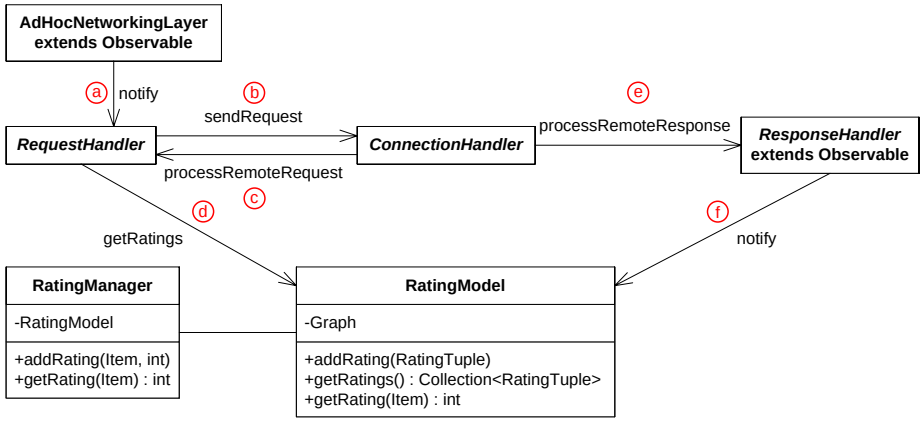

Fig. 5. UML diagrams of CF classes and workflow.

The interaction of the components that perform collaborative filtering if a new remote peer appears in the vicinity of a local peer is indicated in Fig. 5. The local request handler is a registered listener of the ad-hoc networking layer. It is notified whenever a new peer has entered the area of reachability ⓐ. As a result, the request handler issues a request using the connection handler ⓑ. However, this request is not sent immediately but rather queued for a duration p, the parameter introduced in Sect. 3. Consequently, when a request is actually sent, the peer might no longer be reachable and no response will be received. Hence, by delaying the request, we avoid the case of exchanging ratings based on a transient encounter. Eventually, the connection handler receives the response and forwards it to the local response handler ⓔ. The new rating tuples are added to the rating model where they will be available for future rating inference ⓕ. For each response message received, the weight of the edge pointing from the local user to the responding user is increased. At the same time, the remote peer also requests the local peer for rating tuples. Hence, the connection handler receives a request message and forwards it to the remote request handler ⓒ. The request processing consists in returning all ratings made by the local user retrieved from the rating model ⓓ. Note that the local request handler keeps track of peer encounters per consumed item such that ratings are not exchanged multiple times during a single consumption. The number of rating tuples stored in the rating model can be bound to a given size. Therefore, the continuous proximity measure is used to remove the ratings received from the furthest away users.

```
<?xml version="1.0" encoding="UTF-8" ?>
<message>
  <from>Fred Fergusson</from>
  <to>Mary MacBean</to>
  <ratings>
    <rating>
      <item>Pool Of Life</item>
      <value>1</value>
    </rating>
    ...
  </ratings>
</message>
```

Fig. 6. Example response message

Since no information must be provided by a requesting peer for the responding peer to process the request, the request data simply consists of the identifiers of the requesting and requested peers. In contrast, the response data additionally contains a collection of rating tuples. In the scope of this work, requests and responses are sent as plain XML strings without compression or encryption. Figure 6 shows an example response message received by Mary from Fred. It contains ratings made by Fred which are also displayed in the graph in Fig. 2.

5 Conclusion

We have presented a technique to perform user-based CF by exploiting the opportunistic mode of information sharing that results from ad-hoc peer-to-peer networking. Each user's mobile device stores all explicit ratings made by its owner as well as ratings received from other users. Only users in spatio-temporal proximity are able to exchange ratings and we have shown how this provides a natural filtering based on social contexts. An initial version of our CF system has been implemented based on a general peer-to-peer platform developed for the rapid prototyping of peer services.

The idea for this CF approach and the scenario presented in Sect. 2 evolved from an ongoing project to develop mobile information systems for visitors to the Edinburgh festival. In the next stage of this project, the CF system presented in this paper will be integrated with a general interactive festival guide and evaluated at the festival in August 2006. This integration will also involve the incorporation of user reviews and comments. As opposed to numerical rating values, these cannot be aggregated in the manner we have proposed to aggregate ratings. Nevertheless, the CF technique we have presented can also be used to filter reviews and comments.

We also want to investigate how we can extend our CF approach to allow for ratings and reviews exchanged between users in spatial, but not temporal, proximity. In Sect. 3, we introduced the notion of location and periodic event items and the fact that we cannot assume that users consuming such items do so simultaneously. Users consuming these simultaneously were regarded as sharing an additional similarity, namely that they are consuming the item during this particular period of time. However, users consuming the same location item or attending the same periodic event at different times still share a similarity in terms of interest and taste, whereas they do not meet.

References

1. Linden, G., Smith, B., York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing (2003)
2. Edinburgh Festival Fringe. (http://www.edfringe.com/)
3. Sarwar, B., Konstan, J., Borchers, A., Herlocker, J., Miller, B., Riedl, J.: Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In: Proceedings of CSCW '98, ACM Conference on Computer Supported Cooperative Work, Seattle, USA (1998) 345–354
4. Breese, J., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: Proceedings of UAI '98, 14th Conference on Uncertainty in Artificial Intelligence, Madison, USA (1998)
5. Claypool, M., Gokhale, A., Miranda, T., Mumikov, P., Netes, D., Sartin, M.: Combining Content-Based and Collaborative Filters in an Online Newspaper. In: Proceedings of SIGIR '99, Workshop on Recommender Systems, Berkeley, USA (1999)
6. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Item-based Collaborative Filtering Recommendation Algorithms. In: Proceedings of WWW 10, 10th International World Wide Web Conference. (2001) 285–295
7. Robu, V., Poutré, J.L.: Constructing the Structure of Utility Graphs Used in Multi-Item Negotiation through Collaborative Filtering of Aggregate Buyer Preferences. In: Proceedings of RRS 2006, 2nd International Workshop on Rational, Robust and Secure Negotiations in Multi-Agent Systems, Hakodate, Japan (2006)
8. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of CSCW '94, ACM Conference on Computer Supported Cooperative Work, Chapel Hill, USA (1994) 175–186
9. Aggarwal, C.C., Wolf, J.L., Wu, K., Yu, P.S.: Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In: Proceedings of KDD '99. 5th International Conference on Knowledge Discovery and Data Mining, San Diego, USA (1999) 201–212
10. Xu, B., Ouksel, A., Wolfson, O.: Opportunistic Resource Exchange in Inter-Vehicle Ad-Hoc Networks. In: Proceedings of MDM 2004, IEEE International Conference on Mobile Data Management, Berkeley, USA (2004)
11. Anoto AB. (http://www.anoto.com)
12. Belotti, R., Decurtins, C., Norrie, M.C., Signer, B., Vukelja, L.: Experimental Platform for Mobile Information Systems. In: Proceedings of MobiCom 2005, 11th Annual International Conference on Mobile Computing and Networking, Cologne, Germany (2005) 258–269
13. Mirza, B.J., Keller, B.J., Ramakrishnan, N.: Studying Recommendation Algorithms by Graph Analysis (2001)
14. Huang, Z., Chung, W., Ong, T., Chen, H.: A Graph-Based Recommender System for Digital Library. In: Proceedings of JDCL 2002, 2nd Joint Conference on Digital Libraries, Portland, USA (2002) 65–73
15. Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J.C., Pouyoul, E., Yeager, B.: Project JXTA 2.0 Super-Peer Virtual Network. Technical report, Sun Microsystems, Inc. (2003)
16. Apache Axis. (http://ws.apache.org/axis)
17. Java Universal Network/Graph Framework (JUNG). (http://jung.sourceforge.net)