Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Context-Aware Recommendations through Context and Activity Recognition in a Mobile Environment

Toon De Pessemier · Simon Dooms · Luc Martens

Received: date / Accepted: date

Abstract The mobile Internet introduces new opportunities to gain insight in the user's environment, behavior, and activity. This contextual information can be used as an additional information source to improve traditional recommendation algorithms. This paper describes a framework to detect the current context and activity of the user by analyzing data retrieved from different sensors available on mobile devices. The framework can easily be extended to detect custom activities and is built in a generic way to ensure easy integration with other applications. On top of this framework, a recommender system is built to provide users a personalized content offer, consisting of relevant information such as points-of-interest, train schedules, and touristic info, based on the user's current context. An evaluation of the recommender system and the underlying context recognition framework shows that power consumption and data traffic is still within an acceptable range. Users who tested the recommender system via the mobile application confirmed the usability and liked to use it. The recommendations are assessed as effective and help them to discover new places and interesting information.

Keywords Context-Aware · Recommender System · Activity Recognition · Mobile

T. De Pessemier - S. Dooms - L. Martens
Wica, iMinds-Ghent University
G. Crommenlaan 8 box 201, 9050 Ghent, Belgium
Tel.: +32-09-33-14908
Fax: +32-09-33-14899
E-mail: toon.depessemier@ugent.be
E-mail: simon.dooms@gent.be
E-mail: luc1.martens@ugent.be

1 Introduction

Most existing research in the domain of personalized recommendations focuses on suggesting users the most interesting items based on the users' preferences, but without taking into account any additional *contextual information*, such as the user's location, the weather, the time of day, the day of the week, the user's physical activity and mobility, etc. However, the context is an important aspect in the decision process of the user, particularly for mobile applications.

Several studies demonstrated that this contextual information does matter during the selection of information and helps to increase the quality of recommendations [1, 25]. Contextual information can be gathered automatically without interfering with the user's activity by using the built-in sensors of mobile phones, such as the GPS, accelerometer, proximity sensor, compass, etc. Stimulated by the technological evolution of chip design, mobile devices are becoming equipped with better and more sensors. In this respect e.g., Google has introduced a sensor API to handle data from humidity and temperature sensors for its Android operating system [19]. As more contextual information becomes available through a variety of new sensors in mobile devices, the accuracy of context-aware recommender systems can be improved by using these additional knowledge sources.

The aim of this research is twofold. 1) Recognizing the user's basis physical activities as well as more complex contextual situations in a daily user environment. 2) Providing the user personal suggestions for information, based on the recognized activity and context.

In Section 3, we present a framework to recognize the user's physical activity and context based on the sensor data originating from the user's mobile phone. The developed framework first detects basic contexts and activities such as walking and cycling by analyzing the acceleration of the mobile device. By analyzing these basic activities over a longer period of time, recognizing more complex contexts such as "walking to a railway station while it's rainy" is possible, thereby achieving the first aim of this research.

On top of this framework, various applications can be built that use the derived contextual information as input. A typical use case for this contextual information is the context-aware recommender system as discussed in Section 4. This context-aware recommender system provides personalized information and suggestions that are adapted to the current context and activity of the user, thereby achieving the second aim of this research. Finally, the recommender system and the underlying recognition framework are evaluated in terms of power consumption and data traffic. The accuracy and usefulness of the recommendations is assessed via a user study (as discussed in Section 5).

2 Related work

Driven by the increased availability of mobile Internet and the rise of a variety of mobile devices, context-aware applications experienced a growing interest

during the last few decades. A context-aware system uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [9].

Integrating the geographic location of a mobile user has enabled the development of location-based services [29] and context-aware computing is aimed to have applications doing the right thing at the right time and place for users. Location-based services are described by Schiller and Voisard as "services that integrate a mobile device's location or position with other information so as to provide added value to a user" [29]. Context-aware computing was defined by Schilit and Theimer as "software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time" [28].

Nowadays, contextual information is used in many application domains such as multicasting services [2] and media portals [8] to offer users a service that is adapted to their location, needs, and expectations. Also in recommender systems, the user context has gained an increased interest from researchers [1]. For instance, the user's emotions can be used as input for context-aware music recommendations by using support vector machines as emotional state transition classifier [15]. Also the user's social context (age, occupation, location, gender) can be integrated into the recommendation process. Groups of users who are more interesting or important (in terms of demographic characteristics) can receive a higher weight in the collaborative filtering algorithm [24]. In addition, the date, temperature, season, and time of the day can be included as contextual data to further filter the available information relevant to the user's current context [6].

In the application domain of tourism, various applications use the current location of the user to personalize and adapt their content offer to the current user needs [27]. An interesting example is a mobile recommender system proving personal recommendations for Points Of Interest (POIs) based on the user ratings [20]. User ratings can be weighted higher to differentiate between users that rate POIs using the mobile tourist guide application in direct proximity of the POI and others using the Internet away from the POI.

Still, via mobile devices such as smartphones, more contextual information can be retrieved than currently exploited by traditional recommendation algorithms. The context-aware recommender system of Section 4 goes beyond traditional solutions by incorporating not only the location of the user, but all the contextual information that is directly available or can be derived from sensor data or external services. Users are carrying their mobile device on them, resulting in additional information such as their movement, environment, travel speed, etc. This additional information can revolutionize the role of recommender systems from topic oriented information seeking and decision making tools to information discovery and entertaining companions [27].

Additional contextual information can also be acquired by processing raw data originating from sensors. These sensors can be built into mobile devices or even utensils such as a coffee cup to augment them with awareness of their environment and situation as context [12]. By means of sensors in various

kitchen utensils, the activities in the kitchen can be tracked and the user's cooking competence can be estimated [31]. This cooking competence can be used as background knowledge to recommend the user a new recipe.

Nowadays, high-end mobile devices are equipped with many different sensors stimulating the emergence of mobile context-aware applications. To ease the development of these context-aware applications, frameworks have been introduced to provide an abstraction for sensors and actuators. Such a framework assists application developers in gathering data from various sensors, represent application context, and reason efficiently about the context, without the need to write complex code [4]. However, most of these frameworks provide only low-level sensor data and do not interpret the data over a longer period of time to deduce high-level context information such as the user's activities or habits. In contrast, the framework proposed in Section 3 monitors the low-level sensor data, processes these data to deduce basic physical activities or context changes, and analyzes these subsequent activities and context changes to recognize more complex user behavior.

In the literature, various attempts have been made to recognizing the user's physical activity from accelerometer data. Wearable sensors have been used to measure acceleration and angular velocity data in order to recognize and classify sitting, standing, and walking behaviors [23]. An experiment with five biaxial accelerometers worn simultaneously on different parts of the body, showed that it is possible to recognize a variety of different activities like walking, sitting, standing, but also watching TV, running, bicycling, eating, reading etc. [3]. Unfortunately, users have to wear sensors on hip, wrist, arm, ankle, and thigh, which makes this solution less feasible and comfortable for activity recognition in an everyday environment. The recognition performance drops only slightly if data of only two biaxial accelerometers is available - thigh and wrist, but wearing these sensors on the body can still interfere with the user's daily activities.

Also through a single triaxial accelerometer worn near the pelvic region, user activities can be recognized with fairly high accuracy. Nevertheless, experiments showed that activities that are limited to the movement of just hands or mouth (e.g., brushing teeth) are comparatively harder to recognize using a single accelerometer [26]. Moreover a single accelerometer is typically not able to measure ascending and descending stairs walking [22]. Although most mobile devices contain only a single triaxial accelerometer, these results indicate the ability to detect user activities through this built-in accelerometer. The framework proposed in this paper shows (in Section 3.2.8) that it is possible to recognizing physical activities based on data originating from the accelerometer of a mobile phone that is carried in the pocket of the user. By using conventional mobile phones to gather accelerometer data, the physical activity can be recognized transparently, thereby not interfering with the user's normal behavior.

3 Recognition framework

Because of its rapid growth in popularity and widespread use, we opted for Google Android as implementation platform of our framework. Nowadays, almost every Google Android device has several built-in sensors, such as an accelerometer and GPS. But sensor data is also available in many other operating systems for mobile devices.

The context recognition framework consists of three successive phases: 1. *Monitoring* the (sensor) data, i.e. logging the raw data from the accelerometer, GPS, battery, proximity sensor, cell ID, etc. 2. *Processing* the sensor data and recognizing basic activities. 3. *Analyzing* the successive basic activities and recognizing the overall context.

3.1 Monitoring

This phase involves the gathering of all possible raw data from the device. GPS data provides location updates. If no GPS data is available (e.g., in indoor environments), the cell-ID can give an indication of the location through the ID of the cellular tower that is currently providing reception to the device. Further, the battery status (e.g., charging) of the device as well as the battery level can be retrieved. The accelerometer of most Android devices is capable of capturing the device's acceleration on three axes every 20 ms. This accelerometer data is used to recognize the activity of the user. The proximity sensor is used by the Android operating system to detect if an object is in the vicinity of the device. Its main purpose is to detect if the user is holding the device next to the ear for making a phone call. In that case, the screen can be switched off to save power. In this research, the proximity sensor is used to detect where the user carries the device. If the proximity sensor detects no object in the vicinity of the device, then the device is not in the pocket of the user, and recognizing basic activities based on accelerometer data is not reliable. The framework can easily be extended with additional sensor data in order to add additional contextual information.

3.2 Processing

In this phase, each type of data obtained in the monitoring phase, is converted into basic contextual information by a processing unit. For some sensor data, such as data from the proximity sensor, this conversion is straightforward. Other sensor data, such as data from the accelerometer, require a more intelligent processing to obtain contextual information. If additional sensor data become available, the framework can be extended with a new processing unit to extract valuable information from it.

3.2.1 Points-of-interest

Matching the current location of the user to the location of a Point Of Interest (POI) enables the framework to identify the nearest POIs or the POIs within a specified range. The location of the user is retrieved via GPS data or (if GPS is not available, or switched off) estimated by the current cell-ID. Different services are used to retrieve data about the POIs in the current neighborhood of the user. The location of the Belgian railway stations is retrieved via the iRail API [30], a service that provides information about railway stations, schedules, and delays in Belgium. Via the Foursquare API [11] and the Google Places API [14], the framework retrieves data about various other types of POIs such as restaurants, bars, shops, etc.

3.2.2 Urbanization

The POIs that are retrieved by the Foursquare API are used to estimate the urbanization of the current location of the user. The more POIs in the neighborhood of the device, the higher the urbanization level of the neighborhood.

3.2.3 Weather

To find out the weather conditions, the location of the user is first converted into an address via the Google Geocoding API [13]. Subsequently, the ZIP code of the address is used to retrieve weather information from the Google Weather API. This information is refreshed after a change in location or if more than 2 hours have elapsed. To retrieve data about the current weather and urbanization level, GPS data are not strictly required since an estimation of the location of the device by the cell-ID is sufficiently accurate.

3.2.4 Movement

Based on location updates of the GPS data (or cell-ID info) and the coupled timestamps, the framework calculates the current speed and future position of the user. Together with the information about the POIs, the framework can detect if the user is approaching a POI.

3.2.5 Company

In the application, users can add other users as friends and specify their relationship with these friends, e.g., husband, child, buddy etc. Besides, users can opt to share their location data in order to enable the framework to detect whether different users are in another's company or whether some of their friends are in the neighborhood. To ensure the privacy of the user, this is a feature that users can turn on or off, depending on the situation.

3.2.6 Available time

By checking the user's appointments in the calendar application of the phone, the framework can estimate the availability of the user. Appointments in the near future can influence the behavior of the user. E.g., if the user has an appointment within one hour, (s)he might choose a nearby restaurant to have lunch.

3.2.7 Battery

Information about the status of the battery can be used to deduce contextual information about the user, e.g., charging the battery indicates a fixed position of the user. (Many users charge their phone while they are at home.) Data about the battery level can be used to decide to switch off the framework to extend the battery lifetime.

3.2.8 Physical activity

Recognizing physical activities based on patterns in the data originating from the accelerometer is the most complicated processing task of the framework. The framework tries to distinguish four basic activities: standing still, walking, running, and cycling. These different activities induce different accelerations along the three dimensions (X-axis, Y-axis, and Z-axis); and these patterns in the accelerometer data are used to distinguish the basic activities. An important requirement is that users have to carry the mobile device in their pocket, so that the movement of the user's leg can be registered by the device.

Learning to recognize patterns in the accelerometer data is done by training the framework with samples of real physical activities. To obtain these training data, accelerometer data from 11 different users (between 16 and 50 years old) performing the four activities was collected. Every user was asked to perform one of the basic activities during a 5-seconds time frame while a mobile device recorded the accelerometer data. This was repeated for all four basis activities, thereby yielding 44 training samples. This training data clearly showed different patterns for the four activities. E.g., standing still induces the least activity on the accelerometer, cycling produces a data pattern with a periodic variation in time, and running shows more energy than walking.

These training data were used for determining the five discriminating features based on which the four basic activities are distinguished:

1. The average resultant acceleration, i.e. the average of the square root of the sum of the values of each axis squared $\sqrt{x_i^2 + y_i^2 + z_i^2}$.
2. The difference between maximum and minimum acceleration (for each axis).
3. The average deviation to the mean (for each axis), i.e. the average of the absolute difference between a measured sample of the acceleration and the mean acceleration.

4. The sum of the squared deviations to the mean value (for each axis), i.e. the sum of the squared differences between a measured value of the acceleration and the mean acceleration.
5. The deviation of the acceleration (for each axis), i.e. the average of the absolute difference between a measured sample of the acceleration and the sample measured after three time units (so after 60ms).

The first three of these discriminating features were also identified in related work with respect to activity recognition on mobile devices [21]. Discriminating feature (4) and (5) help to distinguish the basic activities based on typical characteristics such as the required energy for the activity and the variation of the acceleration in time.

Based on these discriminating features, newly-acquired accelerometer data can be classified into one of the basic activities. This classification task is performed by using Support Vector Machines (SVM) with an RBF-kernel. Using cross validation thereby considering the data from 1 user as test data and the data from the other users to train the model, each of the 44 logged activities could be classified correctly by the SVM model.

3.2.9 Proximity

As explained in Section 3.1, the data of the proximity sensor can indicate that the device is not in the pocket of the user. Since the recognition of physical activities requires the user to carry the device in his/her pocket, this proximity data can indicate if the activity recognition is reliable.

3.3 Analyzing

Based on the basic activities that are recognized by processing the accelerometer data and the additional contextual information gathered in the processing phase, the framework can recognize more complex user behavior. The underlying idea of the analyzing phase is that complex user behavior consists of different basic contexts which have some relation with each other. E.g., "The user is walking home while it's rainy" consists of "The user is walking", "The user is approaching his/her house" and "it's rainy". The common conditional relationship between these basic contexts is the timing; they have to occur at the same time.

So to recognize complex user behavior, these complex activities are first decomposed into different basic contexts that have a conditional relationship to each other. A basic context can be: the current weather, the current time and day, the battery status and level, being located in an urbanized area, being located in the neighborhood of a specific POI, approaching a specific POI, being in the company of another user, traveling with a specific speed (range), the distance traveled in a specific time interval, or a physical activity such as standing still, walking, running, or cycling. For each potential complex activity, the framework checks if the first basic context matches the data that

are gathered in the processing phase. If this is the case, the framework checks the conditional relationship of this basic context to the second basic context. The conditional relationship can indicate that the second basic context has to occur in parallel with the first basic context or within a specified time frame (e.g., within the next 60 minutes after the first context was detected). So upon detecting the first context, until the conditional time frame has elapsed, the framework monitors the sensor data and tests if the processed data match the pattern of the second basic context. This procedure of matching the processed sensor data to the basic contexts and testing the conditions, is repeated for all basic contexts and conditions of the complex activity.

As soon as one of the basic contexts of the complex activity cannot be matched to the processed sensor data or one of the conditions between the basic activities is not met, the complex activity cannot be recognized. Only if all basic contexts are recognized and all conditions are met, the complex activity is flagged as recognized.

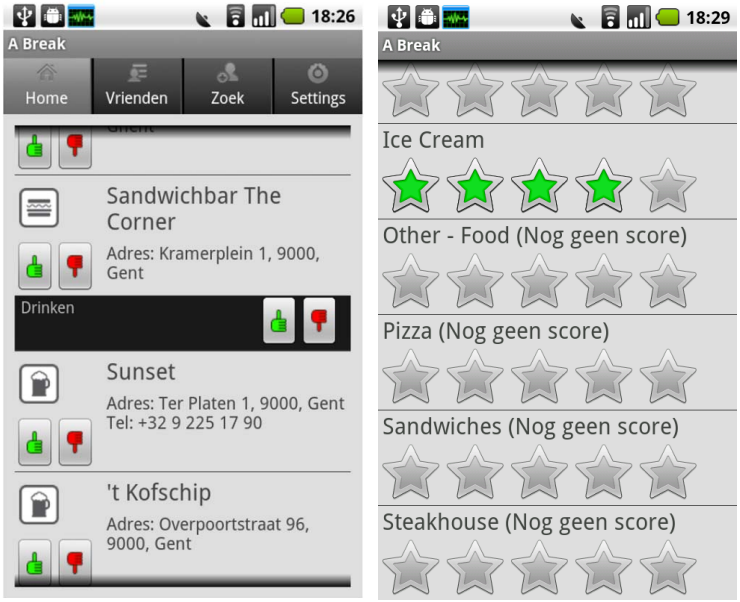
An example of a complex activity is "taking the train" which is composed of the following subsequent basic contexts: 1) The user is approaching a railway station. 2) The user is in the neighborhood of a railway station. 3) The GPS connection may be disconnected. (Although GPS data are available inside a car, GPS data are in many cases not available inside the train or in a building such as the railway station.) 4) The user is traveling with a minimum speed. (In case GPS info is not available, location updates are based on the cell-ID.) 5) In parallel with 4), the user is traveling in the direction of another (nearby) railway station. As soon as these basic contexts and conditions are recognized, the framework believes that the user is traveling by train. This complex activity does not include the act of arriving at the railway station of the destination. If the destination would be included in the complex activity, then the activity could only be recognized after the train journey. Nevertheless, for many applications such as personalized information and recommendations, the recognition has to be performed as soon as possible during the user activity.

In the current implementation, a set of complex activities is defined, but depending on the use case, the framework can also be extended with new complex activities by composing existing or new basic contexts and conditions.

4 Context-aware recommendations

Based on the contextual information that is provided by the context recognition framework, we developed a context-aware recommender system that offers personalized information according to the preferences and current context of the user. Figure 1 shows two screenshots of the user interface of the mobile application through which the users receive their recommendations.

To enable the use of community knowledge (i.e. data regarding user behavior, feedback, and contextual information from all users of the system) in the recommendation process, personalized recommendations are calculated on a centralized server. Based on this community knowledge, Collaborative Filter-



(a) (b)

Fig. 1 Screenshots of the mobile application, showing the list of recommended information items (a) and the possibility to provide explicit feedback (b).

ing (CF) techniques can be used to assist in the recommendation process [5]. The mobile client can send the server a request for recommendations combined with the current context that is retrieved from the context recognition framework. Based on the stored preferences of the user and the contextual information, the server calculates the most appropriate context-aware recommendations tailored to the user's (current) needs.

As shown in Figure 2, the recommendation process consists of three successive phases:

1. *Determining the categories* of information that are most suitable according to the current context of the user.
2. *Fetching the information* of the items of these selected categories.
3. *Selecting the most suitable items* from the retrieved information according to the context and preferences of the users.

After determining the categories and selecting the items, an aggregator combines the partial results.

4.1 Determining the categories

In the first phase, the recommender system receives the current context of the user as input, and predicts the information categories that match this

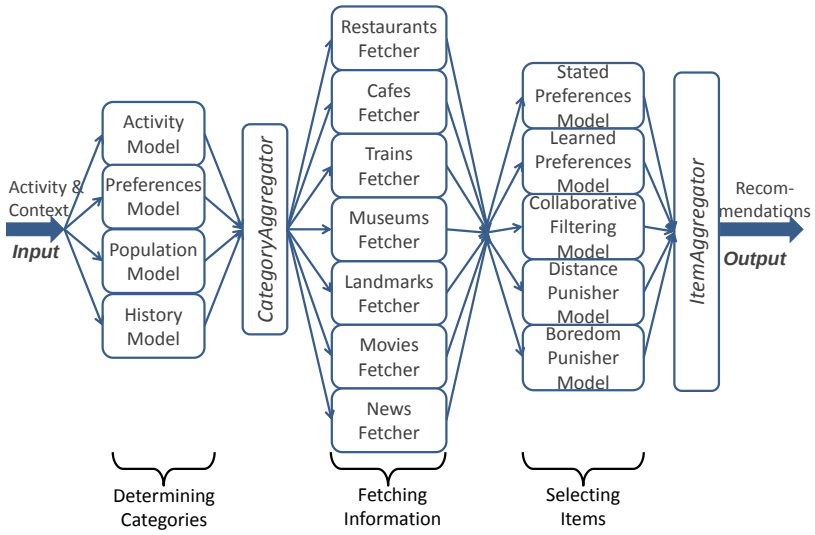


Fig. 2 Schematic overview of the recommendation process, which consists of three successive phases: determining the categories, fetching the information, and selecting the items.

context. One obvious example: if the user is approaching a railway station, information regarding the train schedule might be interesting for the user. To determine the suitability of an information category, four information models work together: the activity model, preferences model, popularity model, and history model. Each of these models assigns a probability score to each information category. This score estimates the conditional probability that the user is interested in information of the specific category, given the current context of the user. The information categories that are used are Food (restaurants, bakeries, bars, etc.), Movies (schedules, descriptions, etc.), Trains (schedules, railway stations, delays, etc.), Monuments (info about churches, statues, etc.), and News (newspaper articles, RSS feeds, etc.); but the system can easily be extended with other categories.

4.1.1 Activity model

The activity model is a knowledge-based system, consisting of a set of general rules that apply to all users. These rules connect a context to an information category that may be interesting for the user in that context. E.g., the context "being in a new city" and "sunny weather" is linked to the information category "Monuments", since users might be interested to do some sightseeing if the weather is good. The context "Evening" is linked to the information category "Food", since information about restaurants for having dinner might be interesting. But the activity model can also contain restrictions, i.e. rules specifying that some information categories cannot be recommended to a spe-

cific group of users in a specific context. E.g., underage users, who are not allowed to drink alcohol, will never receive a recommendation for a bar.

All these rules are stored as triplets (context, category, score), in which the score estimates the probability that users are generally interested in a category, given the specified context. These general trends of the activity model also provide a solution to the cold start problem, the initial situation in which no information about the preferences of the user is available. If no personal preferences are known, the user receives recommendations based on the knowledge of these general trends.

4.1.2 Preferences model

The activity model defines rules for the whole community; but via the preferences model, rules can be specified for each individual user. This way, user preferences for a specific information category, given a specific context, can be specified. E.g., user "Alice" always wants to receive items of the category "News", if she is traveling by train in the morning. The preference model can also help to refine or correct the general rules or assumptions made in the activity model. E.g., users that are approaching the railway station are probably interested in the train schedule. However, people who work at the railway station or live nearby are not interested. The preference model can contain a specific rule for these people who are not interested in information regarding the trains.

These personal rules are stored as 4-tuples (user, context, category, score), in which the score indicates how important this rule is for the user. An initial explicit questionnaire can be used as input to compose these rules.

4.1.3 Popularity model

This model keeps track of the historical behavior of users and learns in which information categories users are interested, given the context. This self-learning process is based on the feedback that users can provide for information categories. Figure 1(b) shows a screenshot of the user interface of the mobile application illustrating the possibility to provide feedback. The popularity model collects feedback information from all users to discover general relations between a context and an information category. The result of this model is a set of triplets (context, category, score) in which the score estimates the probability that users are generally interested in a category, given the specified context. The more users and the more often these users have provided positive/negative feedback on a content item of a specific category in a specific context, the higher/lower the score.

4.1.4 History model

In contrast to the popularity model, which learns category preferences for different contexts on the community level, the history model learns category

preferences for each context on a user level. The model aggregates the historical behavior of each user into a profile to learn the user's personal practices and habits. E.g., user Alice may be interested in the train schedule as soon as she leaves her home in the morning. So, the history model calculates for every user, context, and category, a score which estimates the probability that a specific user is interested in a category, given the context. The more often the user has provided positive/negative feedback on a content item of a specific category in a specific context, the higher/lower the score. These personal habits are stored as 4-tuples (user, context, category, score).

4.1.5 Aggregating the category scores

Each of the models generates its own score, which estimates the probability that the user is interested in a specific category, given the specified context. To obtain a single probability value for each category, the individual scores are normalized and aggregated using a weighted average. In the current implementation, the weights are fixed and set to prioritize the models that reason based on data of the individual users, i.e. the preferences and history model. However, one can argue that varying weights might be more efficient: assigning more importance to the models that are based on community knowledge if a limited amount of personal preferences are available; and raising the importance of the models that use individual data if more knowledge about the user's individual preferences becomes available.

The resulting scores determine the importance of each information category for the user. Therefore, the user receives a proportional number of items of a specific information category as recommendations. Items of an information category with a high score are more common in the recommendation list, whereas items of an information category with a low score are rare or even not present in this list.

4.2 Fetching the information

As soon as the most suitable categories are determined, given the preferences and context of the user, information items belonging to these categories can be fetched. Because this information has to be up-to-date (e.g., for the train schedule or newspaper articles) and because this information is dependent on the context of the user (e.g., the neighboring POIs are determined based on the current location of the user), the information items are retrieved at the moment of requesting the recommendations.

Various services are used to fetch information items of the different information categories. Information regarding locations or POIs (e.g., information about monuments, restaurants, shops, bars, trains, etc.), is selected based on the current location of the user. Potentially interesting data about nearby railway stations, train schedules, and delays is retrieved via iRail [30]. Information about POIs in the current neighborhood of the user is retrieved via the

Foursquare API [11] and the Google Places API [14]. WikiLocation is the service that is used for additional information about monuments and landmarks that might be interesting for the user [10].

Information about (cultural) events is available through the service of CultuurNet Vlaanderen [7]. CultuurNet gathers all information about cultural activities, movies, and events in Flanders (i.e. the Northern part of Belgium). This service is used to retrieve e.g., information about movie theaters and the scheduled movies.

Various comparable services that offer news feeds exist. Because of its structured metadata, the RSS feed of HLN [17] is used to obtain the latest news articles of different categories such as sports, business, local news, international news, etc.

4.3 Selecting items

The last phase of the recommendation process is to select the most appropriate items from the fetched information of the relevant categories. To accomplish this task, five models for selecting items cooperate: the collaborative filtering model, stated preferences model, learned preferences model, distance punisher model, and boredom punisher model. Each of these models assigns a score for the usefulness to each item, thereby indicating how interesting or important the item is for the user. Some of these models consider the preferences of the user, whereas others are merely based on the current context of the user.

4.3.1 Stated preferences model

Through explicit feedback for an item or an attribute of the item, users can state their preference for a particular item (e.g., the user's favorite restaurant) or for a set of items characterized by the attribute they have in common (e.g., all Italian restaurants). In the user's profile, explicit feedback for an item propagates to the attributes and the category of the item. E.g., a positive evaluation of a news article about soccer induces a positive assessment for the attributes "Sports" and "Soccer" as well as for the category "News".

As shown in Figure 1(b), users can specify these preferences via a star-rating mechanism in the user interface, thereby creating a personal profile consisting of triplets (user, item or attribute, score). Based on this explicit profile, the stated preferences model assigns a score to each candidate item by considering the user's rating for the item and/or the attributes describing the item.

The context is not considered in the stated preferences model because of the large number of combinations of context and attribute. Specifying preferences for all these different context-attribute combinations can put a heavy burden on the user.

4.3.2 Learned preferences model

Whereas the stated preferences model is based on explicit preferences for items and attributes of items, the learned preferences model extracts these preferences from implicit data and learns the user behavior. By saving the implicit preferences as 4-tuples (user, context, item or attribute, score), this model can also take into account the context of the user. This way, the recommender can learn for example that the user likes fast-food for lunch, a hot soup on a cold winter day, or a soda after running. Also in this model, feedback for an item propagates to the attributes and the category of the item.

Implicit feedback is gathered by tracking the user's location. If the user is approaching a POI, such as a restaurant or a pub, the recognition framework will monitor the time that the user is staying at that POI. Together with the number of visits to the POI, these data provide some insights into the user's preference for the POI. The more a user visits a POI, and the longer (s)he stays there, the better the implicit feedback for that item. In the current implementation, the implicit feedback is a linear function of the duration of a visit and the number of visits, in which the coefficients are determined by the information category of the item. For items of the category "News", implicit feedback is based on the view-time of an article.

4.3.3 Collaborative filtering model

This model predicts a score for each item by using a standard user-based collaborative filtering algorithm, thereby yielding triplets (user, item or attribute, score). Collaborative filtering is a technique to estimate the preferences of a user for not-evaluated items, by using the preferences of many similar users for these items. These similar users are defined as users with similar preferences on a set of previously-evaluated items and are identified by using a similarity metric [5]. Here, the Pearson correlation metric is used for calculating similarities. To enable the calculation of similarities between users, the preferences of all users are gathered and stored in a central server. To ensure the privacy of the users, this personal information is only used by the server to predict a score for each item, and is not revealed to other users of the system. Using the preferences of the community, the collaborative filtering model assigns the highest scores to the items that best match the preferences of the user, but neglects thereby the contextual information.

4.3.4 Distance punisher model

Since the recommender system has to suggest location-based items, such as restaurants, shops, train info, or the cinema schedule, the location of these items with respect to the current location of the user is especially important. The rational behind the distance punisher model is the users' preference for nearby items. E.g., if the user is traveling on foot, faraway places are not attainable and recommendations for these places are undesirable. Therefore,

this model favors items in the direct neighborhood of the user at the expense of more distant places.

The distance that the user is willing to cover in order to reach a POI depends on the travel mode of the user. By bicycle, the user can move faster than on foot; and by car or train, even distant places can be reached. So, the physical activity of the user is important contextual information that is used in the distance punisher model.

Also the weather is a contextual aspect that influences the distance that users are willing to cover. Traveling on foot or by bicycle in combination with snow or rain will strengthen the users' preference for nearby places; whereas in sunny weather conditions, users might like to walk to their destination.

A measure of the accessibility of a place can be obtained by using distance decay curves for the different travel modes. For multiple travel modes and different purposes, the distance decay function fits a negative exponential curve, as demonstrated by research focusing on the detailed relationship between actual travel behavior and the mean distance to various services [18]. However these proposed distance decay functions cannot be adopted in this research (without changes), since the weather is not included as contextual parameter.

So in this research, the usefulness of an item was estimated by a negative exponential function of the distance d, weather w, and physical activity of the user a, as shown by equation 1.

$$usefulness = e^{-f(d,w,a)} \tag{1}$$

Ideally, the function f should be determined based on actual measurements of the distance users travel in the various contexts (i.e. weather conditions in combination with transport modes). However in the current implementation, f is simplified to the product of the distance, a factor determined by the weather, and a factor determined by the travel mode. Table 1 shows the values of these factors for illustration. Faster travel modes and better weather conditions are associated with smaller factors. Smaller factors in combination with the negative exponential curve induce that additional, further located items can also be considered as recommendations.

Also the availability of the user can be a limitation and is therefore checked by this model. Items that are not attainable within the time frame of the user's calendar (i.e., before the next appointment), given the user's transportation mode, are excluded as potential recommendation.

Table 1 The factors that influence the results of the distance punisher model, a factor determined by the travel mode and a factor for the current weather condition.

standing/walking	running	cycling	car/train
10	5	3	1
snow	rain	cloudy	sunny
6	4	2	1

4.3.5 Boredom punisher model

Recommendations should not only reflect the personal preferences of the user (in a specific context), but also help the user to find surprisingly interesting items (s)he might not have otherwise discovered. E.g., recommending the user's favorite restaurant over and over again might be an accurate recommendation but not be useful. In the domain of recommender systems, serendipity is used as a measure of how useful and surprising the recommendations are [16]. To increase the serendipity of the recommendations, the boredom punisher model favors the items that are new for the user at the expense of items that are already explored by the user (i.e. evaluated or selected for more information).

The information category of the item is an important characteristic that is taken into account by the model. The schedule of the movie theater for a movie that the user has already seen and evaluated is not useful, since people normally do not go to the movie theater twice to see the same movie. Likewise, recommendations for news articles that the user has already read are not desirable. In contrast, it might be interesting to provide information on a regular basis about the schedules and delays of a train that the user regularly catches. In conclusion, new, unexplored items receive the maximum score from the boredom punisher model. Items that the user has already interacted with, are disadvantaged by a specified penalty in accordance with the category of the item.

4.3.6 Aggregating the item scores

Each of the models discussed above generates a score that estimates the usefulness of each item based on the current context and preferences of the user. For each item, these scores are then aggregated into a single estimation of the usefulness, which is used to select a subset of the items within each information category as recommendations. Similar to the aggregation of the category scores, the item scores of the individual models are normalized and aggregated using a weighted average. In the current implementation, the weights are fixed and set to prioritize the model that estimates the usefulness based on the explicit preferences of the user, i.e. the stated preferences model.

Also this aggregator can be extended with varying weights to anticipate the development and improvement of user profiles during service usage. For new users of the service who have a limited profile, the stated preferences and distance punisher model may be the most consistent models. As a user utilizes the recommendation service more often, his/her profile becomes more detailed, and as a result, the collaborative filtering, learned preferences, and boredom punisher model are able to make a valuable contribution. So the weights associated to the models can be made variable in accordance with the advancement of the user profile in order to generate more accurate recommendations.

So to conclude, the category score determines the importance of an information category and the corresponding amount of slots for that category in the

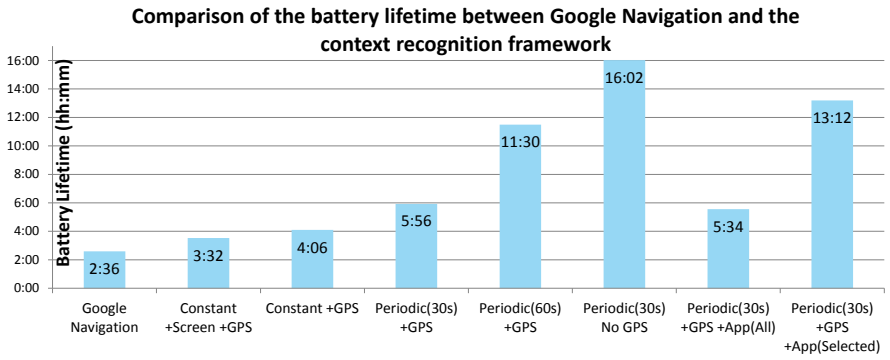


Fig. 3 Evaluation of the battery lifetime while using the context recognition framework

recommendation list. For each information category, the items with the highest estimated usefulness are filling these slots and offered as recommendations to the users, as illustrated in Figure 1(a).

5 Evaluation

5.1 Impact on battery lifetime

Since the proposed framework (as discussed in Section 3) regularly monitors data of various sensors to recognize contexts and activities, battery drain can be a serious issue for the device. GPS localization and data communication via WiFi or a cellular network have a significant impact on the battery lifetime. Therefore, some necessary optimizations were performed to prolong the battery lifetime by carefully adapting the monitoring phase.

To evaluate the impact on the battery lifetime, the framework was compared against Google Navigation, an application that also intensively uses GPS and data communication. The results of this comparison are visualized in Figure 3, which shows the battery lifetime of the test device (HTC Desire) with Google Navigation or the context recognition framework as the only active application. While using Google Navigation, the GPS was active all the time for positioning the device. After selecting a destination point, no additional interaction with the device was performed. The screen of the device was constantly active to the display the route that the user has to take.

Since the screen of a mobile device consumes a lot of energy in the active state, Google Navigation was first compared to a version of the framework in which sensor data is continuously monitored and the screen is constantly active. This always-on version of the framework, which is denoted as "Constant +Screen +GPS" in Figure 3, outperforms Google Navigation in terms of battery lifetime. However, the battery lifetime of three and a half hours is still too limited to be acceptable for a context recognition framework that is supposed to run in the background for a whole day.

In contrast to Google Navigation, the screen does not need to be active for a proper functioning of the context recognition framework. So, an obvious optimization is to dim or switch off the screen of the device. This optimization, in which the framework is continuously monitoring sensor data but in which the screen is not active, is denoted as "Constant +GPS" and yields a gain in battery lifetime of more than half an hour.

Instead of continuously monitoring sensor data, the framework can monitor sensor data in short time periods, and be inactive between these monitoring periods. Since physical activities and contextual changes generally take a few minutes to a few hours, continuously monitoring sensor data is not required to recognize these activities or contexts. In the optimization denoted as "Periodic(30s) +GPS", the framework is inactive during a period of 30 seconds between successive monitoring periods (of 5 seconds). This optimization induces a gain in battery lifetime of almost 2 hours compared to the best scenario with continuous monitoring. To further investigate the influence of this periodic monitoring on the battery lifetime, Figure 3 also evaluates the scenario in which the monitoring of sensor data is switched of during 60 seconds after each monitoring period. The results of this optimization ("Periodic(60s) +GPS") show a gain in battery lifetime of 6 hours and 24 minutes compared to the best scenario with continuous monitoring. A small drawback of the periodic logging of sensor data is the delay to detect changes in activity or context.

In order to further optimize the battery lifetime, switching off the GPS can be considered. The GPS of mobile devices consumes a lot of energy, whereas several processing tasks of the framework can be performed without using GPS data for localization, e.g., the physical activity of the user can be recognized based only on the data of the accelerometer. Moreover, the location and movement of the user can roughly be estimated by (changes in) the current cell-ID. Since localization does not have to be accurate for determining the urbanization level, POI in the neighborhood, or the weather, this approximate localization based on cell-ID can be sufficient. The results show that periodic logging of the sensor data (with inactive periods of 30 seconds) combined with a switched-off GPS ("Periodic(30s) No GPS"), can prolong the battery lifetime to more than 16 hours. So for each use case, the accuracy of the GPS to determine the location and movement of the user has to be balanced against the gain in battery lifetime obtained by switching off the GPS.

The goal of the framework is to provide contextual information about the user as input to other mobile applications. Figure 3 compares the framework with periodic logging and GPS ("Periodic(30s) +GPS") against a combination of the framework and an application that uses the context of the user to provide information about the train schedule ("Periodic(30s) +GPS +App(All)"). Bundling the framework with an application that exploits the contextual information might slightly reduce the battery lifetime (22 minutes in case of the example application). Visualizing the information on the screen and additional processing of the data induces the extra energy consumption. However, not all applications require all contextual information, and contextual information may not be required at any time. E.g., an application for providing info re-

garding the train schedule does not need information about the weather, and data about POIs (i.e. the railway stations) are static and not required at any time. If unnecessary contextual data are not monitored and processed by the framework, the battery lifetime can be improved. In case that the framework is optimized for the application that provides the train schedule, the battery lifetime increases with 7 hours and 38 minutes.

So, although the monitoring and processing of sensor data on a mobile device entails a significant battery drain, these results indicate that the framework can be configured in several ways to obtain an acceptable battery lifetime.

5.2 Data traffic

Since the context-aware recommender system (as discussed in Section 4) queries various services during operation and info of different content providers is fetched for the recommender application, mobile data communication is necessary for the proper functioning of the recommender system and the underlying context recognition framework. Given that some mobile subscriptions are charging users based on their data traffic, the recommender application, combined with the framework running on an HTC Desire device, was evaluated on this criteria.

In this evaluation, we distinguished intensive and non-intensive use of the application and measured the data traffic for both scenarios. Intensive use of the application is defined as "very frequently requesting recommendations and providing feedback". The scenario of intensive use is simulated in the context of "walking in a city center", whereby recommendations are requested for the current location, and feedback on one of these recommended items is provided once per minute. The duration of the test was one hour. So during this walk, recommendations are requested 60 times for different districts of the city, and as many times feedback on one of these items is processed. Non-intensive use of the application differs from intensive use by less frequently requesting recommendations and sporadically providing feedback. During the one hour walk, recommendations are requested 5 times and feedback is provided for 3 of these items.

Table 2 shows the average (avg) and standard deviation (std) of the data traffic in download and upload direction for intensive and non-intensive use of the application. These results indicate that even in the case of intensive use of the application, the total data traffic is only 2.29 MB on average. As a result, the data traffic required for the functioning of the application is acceptable and in the range of the data traffic induced by similar mobile applications.

5.3 User evaluation

To evaluate the usefulness and effectiveness of the recommender application and the personal recommendations, a small user evaluation was performed.

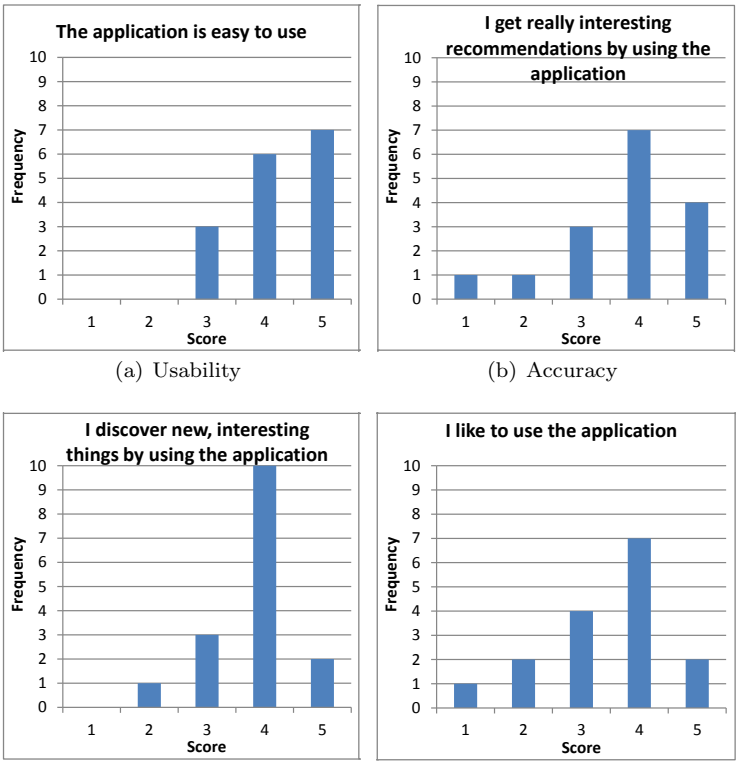
Table 2 Evaluation of the recommender application and the underlying framework in terms of data traffic.

	Intensive	Non-Intensive
Avg Download (MB)	1.97	1.17
Std Download (MB)	0.03	0.08
Avg Upload (MB)	0.32	0.12
Std Upload (MB)	0.02	0.01
Avg Total (MB)	2.29	1.29
Std Total (MB)	0.04	0.08

The test panel consisted of 16 test subjects (12 men and 4 women) who are representative for the target users of the applications. All test subjects were between 21 and 32 years old and make daily use of a smartphone. They were asked to download and install the application on their own smartphone (different brands and types of Android devices) and use it during one week to retrieve recommendations in their daily environment. Since the application has no hard requirements regarding the CPU, memory, or GPU of the device, we believe that asking the users to test the application on their own device can be justified. Moreover, this has the advantage that test subjects are familiar with the device that was used during the evaluation. In addition, the test subjects could use the same device for daily use (texting, phone calls, etc.) and for evaluating the recommender service. To ensure that the test subjects are sufficiently familiar with the application for an evaluation after the test, we asked to use the application at least once a day and at least three times outdoors. The latter requirement stimulates test subjects to use the application on the move, or for exploring new places.

After one week, test subjects received a questionnaire to evaluate the application by means of 9 multiple choice questions and 3 open questions. The multiple choice questions consisted of statements that test subjects had to assess on a 5-point rating scale ranging from "1: totally disagree" to "5: totally agree". The goal of the open questions was to inquire for potential improvements or extensions to the application.

Figure 4 visualizes the answers to the most interesting multiple choice questions as histograms. The first histogram, Figure 4(a), indicates that all test subjects experienced the application as "easy to use". Because of the automatic context recognition and the straightforward way to retrieve recommendations, no test subject provided a negative evaluation regarding the usability. The accuracy of the recommendations is assessed by asking the test subjects if the recommendations are interesting. Except for two people, the test subjects agreed with the statement that the recommendations of the application are really interesting for them, as illustrated in the second histogram, Figure 4(b). The test subject who totally disagreed with this statement had a data connection problem with his mobile phone during the test, which explains why he did not receive (interesting) recommendations. The ability to help users discovering new and interesting information or POIs, i.e. the serendipity of the recommendations, is assessed via the third histogram, Figure 4(c). Except for



(a) Usability (b) Accuracy

(c) Serendipity (d) Likeability

Fig. 4 Histograms of the answers on the multiple choice questions of the user evaluation.

two people (one of them had a connection problem), test subjects confirmed that they can find new and interesting information or POIs via the recommendations. The last histogram, Figure 4(d), gives an indication about how pleasant it is to use the application. Only three test subjects disagreed with the statement "I like to use the application".

Detailed feedback of the test subjects explained their dissatisfaction with the application or the recommendations. According to one test subject, loading the recommendations takes too much time. Two other test subjects would like to have more detailed sub-categories. Besides, two test subjects mentioned the battery drain as a serious drawback. One test subject did not understand the added value of a recommender system for selecting information on a mobile device.

To summarize, the application that offers context-aware recommendations based on the automatically-detected context of the user, is easy to use. According to the test subjects, the personal recommendations are a valuable asset in the context of information retrieval: these recommendations are interesting and help them to discover new content and places.

Via the open questions, test subjects were asked if additional features should be added to the application, and which existing features should be removed. Four test subjects suggested to extend the friend-functionality of the application. Besides adding and removing users from their friend list, they would like to see the context of their friends. They also mentioned the possibility to recommend items to friends and to see their friends' feedback on items. One test subject would like to receive detailed information for additional categories, such as detailed info about the articles in supermarkets. Another feature on the wish list of the test subjects is "changing their own current context". E.g., manually changing the location would be useful to plan a holiday and retrieve the recommendations for the holiday destination before arriving. At last, also a more detailed feedback mechanism consisting of check-ins, likes, and reviews, was mentioned.

Three test subjects indicated that the items of the category "News" might be superfluous. The friend-functionality was also mentioned two times as a feature that can be removed from the application, since it was not clear for the test subjects that this information is used by the recommender.

6 Conclusions

In this research, we investigated how the current context and activity of the user can be recognized based on sensor data and the accelerometer of his/her mobile device. The context recognition framework first monitors and processes the sensor data to recognize basic activities or context changes. Then, these successive basic activities are analyzed to recognize the overall context of the user. An evaluation of the framework proved that physical activities and the context of the user can be recognized with a high accuracy and that this contextual information can be valuable knowledge for a context-aware recommender system. Besides, the framework can be used for other applications, e.g., for monitoring the physical activities of the user in the context of health care.

Several challenges, such as the battery lifetime and the data traffic, are associated with the development of a context recognition framework and a context-aware recommender system that works on top of it. Experiments demonstrated that the battery lifetime of the framework can be improved by switching off the GPS at the expense of an acceptable loss in accuracy regarding the location and movement of the user. Furthermore, even in the case of intensive use, the data traffic generated by the framework and the recommender application on top of it is limited to a couple of MB per hour by constraining the recommendations to the current context of the user.

A user study showed that context-aware recommendations are effective and helpful for discovering new places and interesting information. Moreover, user like to receive information tailored to their current needs and consider the recommender application as easy to use. These results confirm the necessity

to adapt (mobile) applications and service to the activity and context of the user in order to improve their effectiveness and the user experience.

Acknowledgements The authors would like to thank Bart Matté and Ewout Meyns for their programming work in this research project.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)
2. Antoniou, J., Pinto, F., Simoes, J., Pitsillides, A.: Supporting context-aware multiparty sessions in heterogeneous mobile networks. Mobile Networks and Applications **15**, 831–844 (2010)
3. Bao, L., Intille, S.: Activity recognition from user-annotated acceleration data. In: A. Ferscha, F. Mattern (eds.) Pervasive Computing, *Lecture Notes in Computer Science*, vol. 3001, pp. 1–17. Springer Berlin / Heidelberg (2004)
4. Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. In: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), PERCOM '04, pp. 361–365. IEEE Computer Society, Washington, DC, USA (2004)
5. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, UAI'98, pp. 43–52. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
6. Brown, P., Bovey, J., Chen, X.: Context-aware applications: from the laboratory to the marketplace. Personal Communications, IEEE **4**(5), 58–64 (1997)
7. CultuurNet-Vlaanderen: Uitdatabank developer tools (2012). Available at http://tools.uitdatabank.be/docs
8. Debaty, P., Goddi, P., Vorbau, A.: Integrating the physical world with the web to enable context-enhanced mobile services. Mobile Networks and Applications **10**(4), 385–394 (2005)
9. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing **5**(1), 4–7 (2001)
10. Dodson, B.: Wikilocation (2012). Available at http://wikilocation.org/
11. Foursquare: Foursquare API (2012). Available at https://developer.foursquare.com/
12. Gellersen, H., Schmidt, A., Beigl, M.: Multi-sensor context-awareness in mobile devices and smart artifacts. Mobile Networks and Applications **7**, 341–351 (2002)
13. Google: Geocoding API (2012). Available at https://developers.google.com/maps/documentation/geocoding/
14. Google: Places API (2012). Available at https://developers.google.com/places/documentation/
15. Han, B.J., Rho, S., Jun, S., Hwang, E.: Music emotion classification and context-based music recommendation. Multimedia Tools Appl. **47**(3), 433–460 (2010)
16. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)
17. HLN: Rss news feed (2012). Available at http://www.hln.be/rss.xml
18. Iacono, M., Krizek, K., El-Geneidy, A.: Access to destinations: How close is close enough? estimating accurate distance decay functions for multiple modes and different purposes. Tech. rep., University of Minnesota, Twin Cities. Minnesota Department of Transportation (2008). Ref.: MN/RC 2008-11
19. Inc., S.: Humidity and Temperature Sensor for Mobile Devices (2012). Available at http://news.thomasnet.com/companystory/Humidity-and-Temperature-Sensor-for-Mobile-Devices-851215
20. Kenteris, M., Gavalas, D., Mpitziopoulos, A.: A mobile tourism recommender system. In: Proceedings of the The IEEE symposium on Computers and Communications, ISCC '10, pp. 840–845. IEEE Computer Society, Washington, DC, USA (2010)

21. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl. **12**(2), 74–82 (2011)
22. Lee, M.h., Kim, J., Kim, K., Lee, I., Jee, S.H., Yoo, S.K.: Physical activity recognition using a single tri-axis accelerometer. In: Proceedings of the World Congress on Engineering and Computer Science, vol. 1 (2009)
23. Lee, S.W., Mase, K.: Activity and location recognition using wearable sensors. Pervasive Computing, IEEE **1**(3), 24 –32 (2002)
24. Oh, J.M., Moon, N.: User-selectable interactive recommendation system in mobile environment. Multimedia Tools and Applications **57**, 295–313 (2012)
25. Oku, K., Nakajima, S., Miyazaki, J., Uemura, S.: Context-aware svm for context-dependent information recommendation. In: Mobile Data Management, 2006. MDM 2006. 7th International Conference on, pp. 109–109 (2006)
26. Ravi, N., Dandekar, N., Mysore, P., Littman, M.L.: Activity recognition from accelerometer data. In: Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3, IAAI'05, pp. 1541–1546. AAAI Press (2005)
27. Ricci, F.: Mobile recommender systems. Information Technology & Tourism **12**(3), 205–231 (2010)
28. Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. Network, IEEE **8**(5), 22–32 (1994)
29. Schiller, J., Voisard, A.: Location-based services. Morgan Kaufmann (2004)
30. Tiete, Y., Schmitz, S., Colpaert, P.: iRail API (2012). Available at http://project.irail.be/wiki/API/APIv1
31. Wagner, J., Geleijnse, G., van Halteren, A.: Guidance and support for healthy food preparation in an augmented kitchen. In: Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation, CaRR '11, pp. 47–50. ACM, New York, NY, USA (2011)