Context and Activity Recognition for Personalized Mobile Recommendations

Toon De Pessemier, Simon Dooms, Kris Vanhecke, Bart Matté, Ewout Meyns, and Luc Martens

iMinds - WiCa - Ghent University, Dept. of Information Technology, Gaston Crommenlaan 8 box 201, B-9050 Ghent, Belgium
{toon.depessemier,simon.dooms,kris.vanhecke,luc.martens}@intec.ugent.be

Abstract. Through the use of mobile devices, contextual information about users can be derived to use as an additional information source for traditional recommendation algorithms. This paper presents a framework for detecting the context and activity of users by analyzing sensor data of a mobile device. The recognized activity and context serves as input for a recommender system, which is built on top of the framework. Through context-aware recommendations, users receive a personalized content offer, consisting of relevant information such as points-of-interest, train schedules, and touristic info. An evaluation of the recommender system and the underlying context-recognition framework demonstrates the impact of the response times of external information providers. The data traffic on the mobile device required for the recommendations shows to be limited. A user evaluation confirms the usability and attractiveness of the recommender. The recommendations are experienced as effective and useful for discovering new venues and relevant information.

Keywords: Context, Activity Recognition, Mobile, Recommendation, Personalization

1 Introduction

Contextual information is used in many application domains 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, various tourist guide applications use the location of the user to personalize and adapt their content offer to the current user needs. An interesting example is a mobile recommender system proving personal recommendations for Points Of Interest (POI) based on the user ratings [12]. User ratings can be weighted higher to differentiate between users that rate POI 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. Users are carrying their mobile device on them, resulting in additional information such as their location, speed, environment, 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 [16].

Various attempts have been made to recognize user 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 behavior [14]. 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. [2]. Moreover, the recognition performance drops only slightly if data of only two biaxial accelerometers are available - thigh and wrist. 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 [15]. Although most mobile devices contain only a single triaxial accelerometer, these results indicate the ability to detect user activities through this built-in accelerometer.

In this research, we present a framework for recognizing the user's context and activity based on sensor data originating from the user's mobile phone in a daily user environment. The developed framework (Section 2) 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 station while it is rainy", is possible. This contextual information is used by the recommender system (Section 3) in order to achieve the main goal of this research: providing personalized information and suggestions that are adapted to the current context and activity of the user. The response time of the information providers as well as the data traffic required for the recommendations are evaluated. The accuracy and usefulness of the recommendations is assessed via a user study (Section 4).

2 Context and Activity-Recognition

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 are 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.

2.1 Monitoring

This phase involves the gathering of all available raw data from the device. GPS data provides location updates. If no GPS data are 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 are 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.

2.2 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.

Points-of-Interest. Matching the current location of the user to the location of POI enables the framework to identify the nearest POI or the POI 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 POI in the current neighborhood of the user. E.g., the location of the Belgian railway stations is retrieved via the iRail API [17], a service that provides information about railway stations, schedules, and delays in Belgium. Via the Foursquare API [6], the framework retrieves data about various other types of POI such as restaurants, bars, shops, etc.

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

Weather. To find out the weather conditions, the location of the user is first converted into an address via the Google Geocoding API [7]. 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.

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 POI, the framework can detect if the user is approaching a POI. For this detection, the framework considers the type of POI, the direction of the movement, the distance between the user and the POI at successive times, and the movement speed to estimate to which POI the user is on the way.

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.

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.

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.

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 were 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. These training data clearly showed different patterns for the four activities, as shown in Figure 1. 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:

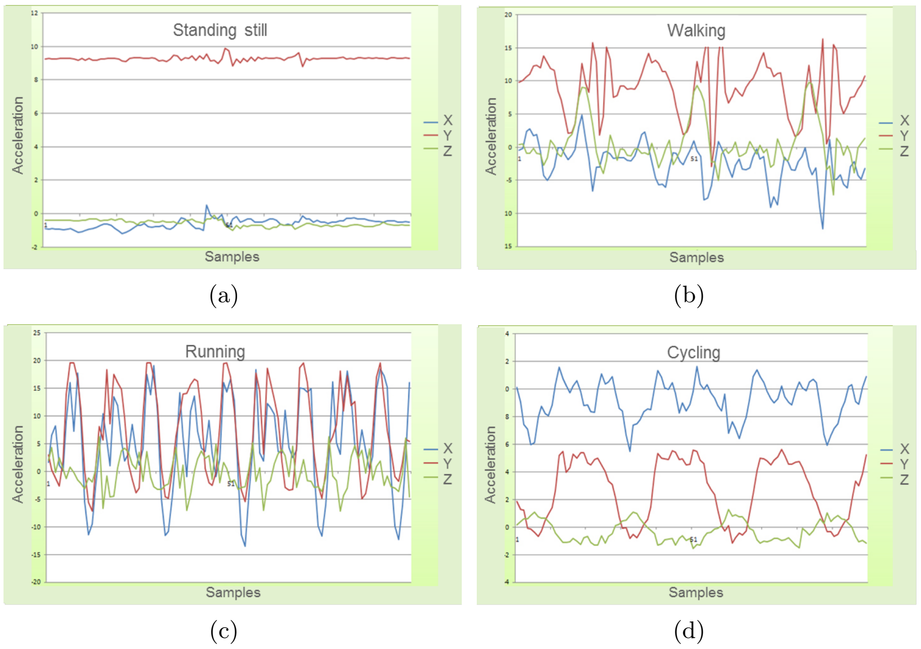


Fig. 1. The data of the accelerometer obtained while doing physical activities.

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 60 ms).

The first three of these discriminating features were also identified in related work with respect to activity recognition on mobile devices [13]. 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.

Proximity. As explained in Section 2.1, 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.

2.3 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) GPS connection is lost. Although GPS data are available inside a car, GPS data are in most cases not available inside the train. 4) The user is traveling with a minimum speed. In this case, location updates are based on the cell-ID, because GPS info is not available. 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 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.

3 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.

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. *Retrieving the information* of the items of these selected categories.
3. *Recommending 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.

3.1 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 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, etc.), Movies (schedules, descriptions, etc.), Trains (schedules, 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.

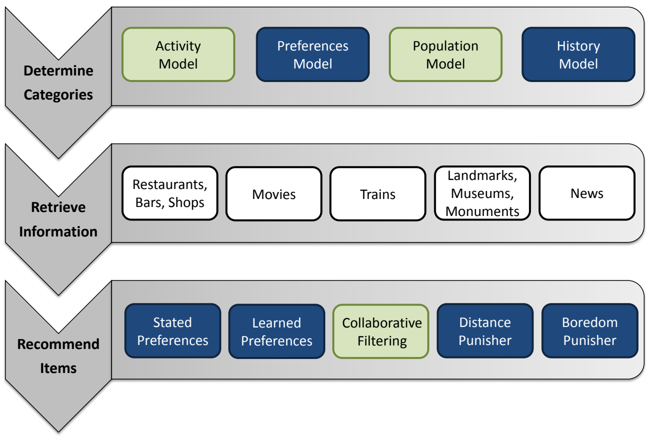


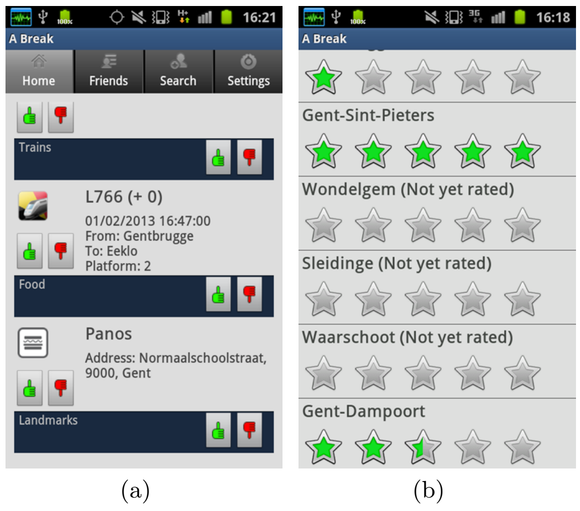
Fig. 2. Schematic overview of the recommendation process, which consists of three successive phases: determining the categories, retrieving the information, and recommending the items. The activity model, population model, and collaborative filter are based on the knowledge of all users of the system; the other models are based on a single user's personal data.

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. 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 offer 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.

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. 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.

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 learning process is based on the feedback that users can provide for information categories. Figure 3 shows two screenshots of the user interface of the mobile application and illustrates 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.



Trains

L766 (+ 0)
01/02/2013 16:47:00
From: Gentbrugge
To: Eeklo
Platform: 2

Food

Panos
Address: Normaalschoolstraat, 9000, Gent

Landmarks

Gent-Sint-Pieters

Wondelgem (Not yet rated)

Sleidinge (Not yet rated)

Waarschoot (Not yet rated)

Gent-Dampoort

Fig. 3. Screenshots of the mobile application, showing the list of recommended information items (a) and the possibility to provide explicit feedback (b).

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. 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).

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. 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.

3.2 Retrieving 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 retrieved. Various services are used to retrieve information items of the different information categories. Information regarding locations or POI (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 are retrieved via iRail [17]. General information about POI in the current neighborhood of the user is retrieved via the Foursquare API [6] and the Google Places API [8]. WikiLocation is the service that is used for additional information about monuments and landmarks that might be interesting for the user [5]. Information about (cultural) events is available through the service of CultuurNet Vlaanderen [4]. 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 [10] is used to obtain the latest news articles of different categories such as sports, business, local news, international news, etc.

3.3 Recommending Items

The last phase of the recommendation process is to select the most appropriate items from the retrieved information of the relevant categories. To accomplish this task, five models for selecting items cooperate: the stated preferences model, learned preferences model, collaborative filtering 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.

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 3, 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.

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 framework will monitor the time that the user is staying at that POI. The hypothesis is that users stay longer at a POI, e.g., a bar, if they are having fun, whereas they leave the POI early if they do not like it. 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 time of the visit and the number of visits, in which the coefficients are determined by the information category of the item. These coefficients specify for instance that spending time in a shop has a different impact than spending time in a railway station. For items of the category “News”, implicit feedback is based on the view-time of an article.

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 [3]. Here, the Pearson correlation metric is used for calculating similarities. 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.

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 [11]. 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 user 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 recommendations.

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

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

not be useful. In the domain of recommender systems, serendipity is used as a measure of how useful and surprising the recommendations are [9]. 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.

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. So to conclude, the category score determines the importance of an information category and the corresponding amount of slots for that category in the recommendation list. For each information category, the items with the highest estimated usefulness are filling these slots and offered as recommendations to the users.

4 Evaluation

4.1 Response Time

The response time of the recommender system is dependent on the available processing power and can be improved by hardware upgrades or parallelization of the calculations. However since the system queries various services during operation and information of different content providers is retrieved for the recommendations, the response time of the system is also strongly influenced by the response times of these information providers, which are beyond our control. Therefore, the response times of the various information providers was evaluated by means of 3000 measurements at different times of the day.

Table 2 shows for all the information providers the type of information they offer, the mean response time, and the standard deviation on the measurements. Foursquare showed to be the fastest information provider, but also news of HLN can be retrieved with a short response time. The response times of CultuurNet and WikiLocation are longer than the response times of Foursquare and HLN but still less than 1.5 seconds and so acceptable for the recommender system. The slowest information provider is iRail with a mean response time of almost 6 seconds for retrieving information about train schedules, railway stations, and delays. The large standard deviation (approximately 11 seconds) illustrates that the response times of iRail are highly varying with peaks up to 30 seconds.

Since the recommender system relies on these information providers, the response times of these information providers directly influence the response time of the recommender system. Caching data can be a partly solution for static information about bars, shops, restaurants, landmarks, and monuments but is not appropriate for rapidly changing information such as train delays and movie schedules.

Table 2. Response times of the different information providers.

Provider	Type of Info	Mean Response Time (ms)	Standard Deviation
Foursquare	Bars, Shops, Restaurants	144	169
HLN	News	201	296
CultuurNet	Events, Movies	861	217
WikiLocation	Landmarks, Monuments	1450	614
iRail	Trains	5810	11028

4.2 Data Traffic

Mobile data communication is necessary for the proper functioning of the recommender system and the underlying context-recognition framework, e.g., for retrieving information about the POI. Given that some mobile subscriptions are charging users based on their data traffic, the recommender application (combined with the framework) 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 3 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 results, 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.

Table 3. Evaluation of the recommendation 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

4.3 User Evaluation

To evaluate the usefulness and effectiveness of the application and the personal recommendations, a small user evaluation was performed. 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 and use it during one week to retrieve recommendations in their daily environment.

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 second histogram, Figure 4(b), 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.

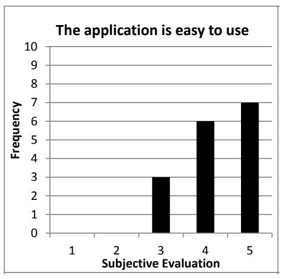
The accuracy of the recommendations is assessed by asking the test subjects if the recommendations are interesting. Except for two people, the test subjects agreed with the statement that the recommendations of the application are really interesting for them, as illustrated in the third histogram, Figure 4(c). The test subject who totally disagreed with this statement had a data connection problem during the test, which explains why he did not receive (interesting) recommendations. The ability to help users discovering new and interesting information or POI, i.e., the serendipity of the recommendations, is assessed via the last histogram, Figure 4(d). Except for two people (one of them had a connection problem), test subjects confirmed that they can find new and interesting information or POI via the recommendations.

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.

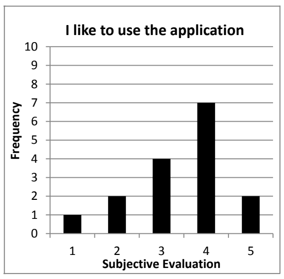
Three test subjects indicated that the items of the category “News” might be superfluous. The friend-functionality was also mentioned two times as a feature that can be removed from the application, since it was not clear for the test subjects that this information is used by the recommender.

5 Conclusions

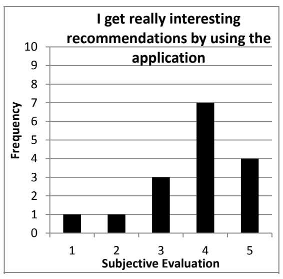
In this research, we investigated how the current context and activity of the user can be recognized based on sensor data and the accelerometer of his/her mobile device. The context-recognition framework first monitors and processes



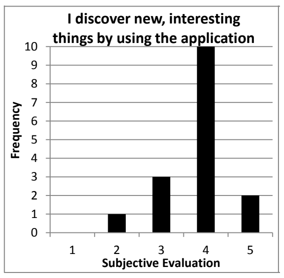
(a) Usability



(b) Attractiveness



(c) Accuracy



(d) Serendipity

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

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 response time and the data traffic, are associated with the development of the context-recognition framework and the recommender on top of it. Experiments demonstrated that the response times of information providers can have a big influence on the response time of the recommender system. The data traffic required for the recommender is limited to a couple of MB per hour, even in the case of intensive use, 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, users 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 services to the activity and context of the user in order to improve the effectiveness and the user experience.

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 (Jan 2005), http://doi.acm.org/10.1145/1055709.1055714
2. Bao, L., Intille, S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) Pervasive Computing, Lecture Notes in Computer Science, vol. 3001, pp. 1–17. Springer Berlin / Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-24646-6_1
3. 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. pp. 43–52. UAI'98, San Francisco, CA, USA (1998), http://dl.acm.org/citation.cfm?id=2074094.2074100
4. CultuurNet-Vlaanderen: Uitdatabank developer tools (2012), http://tools.uitdatabank.be/docs
5. Dodson, B.: Wikilocation (2012), http://wikilocation.org/
6. Foursquare: Foursquare API (2012), https://developer.foursquare.com/
7. Google: Geocoding API (2012), https://developers.google.com/maps/documentation/geocoding/
8. Google: Places API (2012), https://developers.google.com/places/documentation/
9. 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 (Jan 2004), http://doi.acm.org/10.1145/963770.963772
10. HLN: Rss news feed (2012), http://www.hln.be/rss.xml
11. 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
12. Kenteris, M., Gavalas, D., Mpitziopoulos, A.: A mobile tourism recommender system. In: Proceedings of the The IEEE symposium on Computers and Communications. pp. 840–845. ISCC '10, IEEE Computer Society, Washington, DC, USA (2010), http://dx.doi.org/10.1109/ISCC.2010.5546758
13. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl. 12(2), 74–82 (Mar 2011), http://doi.acm.org/10.1145/1964897.1964918
14. Lee, S.W., Mase, K.: Activity and location recognition using wearable sensors. Pervasive Computing, IEEE 1(3), 24 –32 (july-sept 2002)
15. 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. pp. 1541–1546. IAAI'05, AAAI Press (2005), http://dl.acm.org/citation.cfm?id=1620092.1620107
16. Ricci, F.: Mobile recommender systems. Information Technology & Tourism 12(3), 205–231 (2010), http://www.ingentaconnect.com/content/cog/itt/2010/00000012/00000003/art00002
17. Tiete, Y., Schmitz, S., Colpaert, P.: iRail API (2012), http://project.irail.be/wiki/API/APIv1