Activity Context Representation: Techniques and Languages
AAAI Technical Report WS-12-05

Inference of User Context from GPS Logs for Proactive Recommender Systems

Benjamin Lerchenmueller, Wolfgang Woerndl
Technische Universitaet Muenchen, Germany
{lerchenm, woerndl}@in.tum.de

Abstract

With the increasing popularity of smartphones, the wide availability of mobile Internet and the higher computational power of mobile devices, new types of applications are now possible. It is important to provide a smooth user experience by facilitating the interaction with the device. To do so, the goal of the work is support proactive recommendations on the mobile device. In order to determine the best point in time for a recommendation, various context information needs to be taken into account. One interesting aspect is determining the current user activity, e.g. whether the user is walking or not. In this paper, we present an algorithm that runs online on a smartphone and analyzes the user activity based on GPS data.

Introduction

In the past decade, recommender systems have become very popular, which led to wide variety of available system. In this work, we examine the comparatively new approach of proactive recommender systems (Ricci 2011).

Motivation

Traditional recommender systems follow a request-response pattern that requires the user to express the desire for a recommendation. A contrary approach is taken in this work in order to enable a smoother user experience: the system automatically provides recommendations that may be appropriate in a given situation. In such a proactive recommender system, the system generates recommendations and notifies the user about it without any explicit user request (Woerndl et. al. 2011). For the decision whether a recommendation would be appropriate or not, the current user context has to be assessed. One important aspect is the knowledge about the user's current activity.

Target Scenario and Goal

The following description of the target scenario was the driving idea behind this work: *A tourist is walking through a foreign city doing some sightseeing. After quite some time of discovering the city by foot, the user's smartphone suggests to take a break at a recommendable coffee shop close-by.*

The key property of a proactive recommender system is delivering the best recommendations at the right point. The question of finding the best item(s) has already gained a lot of attention in prior research and thus, various well-working systems for retrieving good recommendations exist. However, figuring out the best point in time for a recommendation has not been deeply examined yet. Therefore, determination when to generate a recommendation and present it to the user is the goal of this research project. A important information needed in the target scenario is knowing whether the user is walking or not, because it might have crucial influence on the need for a break and also on the interruptability of the user. The aim of this paper is to present an algorithm that can run on current smartphone and provides this classification in real-time.

Outline

In the next section, important background information on proactive recommender systems will be given before presenting related work. The section Inferring User Activity will provide deep information about the developed algorithm and a needed logging framework. Subsequently, we describe the acquisition of test data and the evaluation before concluding the paper with a short summary.

Background

Proactive Recommender Systems

As mentioned above, the field of proactive recommender systems did not yet experience a lot of attention in

research. Determining of the best point in time for a recommendation is a non-trivial and an important part of a proactive recommender system. If the recommendation is given at the wrong situation the user might be disrupted or annoyed which might lead to refusal of the system.

In our earlier work, we developed a two-phase model to handle proactivity in mobile recommender systems based on the current context (Woerndl et. al. 2011). In the first phase, the system determines whether or not the current situation warrants a recommendation. This calculation is based on several context categories that are explained in the next paragraph. The second phase deals with evaluating the candidate items. If one or more items are considered good enough in the current context in the second phase, the recommender system communicates it to the user. The first phase is executed periodically in the background. The second phase is only executed when the first phase indicates a promising situation and the corresponding score exceeds a threshold. We have also worked on investigating the user interface and acceptance of proactive recommendations on smartphones (Gallego Vico, Woerndl and Bader 2011), but this study is out of the scope of this paper.

Context can be defined as characterizing the situation of entities that are relevant to the interaction between a user and an application (Dey, Abowd and Salber 2001). When assessing the current situation in the first phase of our model, we are utilizing the following four context categories: 1. *User context*, e.g. the current activity of the user such as "walking" as inferred from sensor data, but also the state of the mobile device, e.g. "flight mode", 2. *Temporal context*, e.g. current time, 3. *Geographic context*, e.g. distance to available points of interest, and 4. *Social context*, e.g. whether the user is alone or not.

The attributes in the context categories are modeled as simple attribute-value pairs but more complex options (see Strang and Linnhoff-Popien, 2004, for example) are possible. The main requirement is that the context attributes can be analyzed in real-time on the mobile device.

In this paper, we are focusing on the user context. For this context category, various information available on current smartphones can be taken into account. We developed a three-component scheme for describing and evaluating the different features of user context. At a given point, each component is evaluated leading to three scores. These scores are combined to yield a global user context score indicating the appropriateness of a recommendation for the given situation based on the user context.

The first component of user context is given by the *user status*. It encapsulates features describing the state of the user, like the telephony state, the presence of calendar entries etc. The *device status* constitutes the second component and is structurally very similar to the user status, but incorporates different features, for instance the state of connectivity. Both components capture the encapsulated features and calculate the recommendation score based on the following idea. For each feature, a value indicating its weight *w* exists. Furthermore, every feature value has an appropriateness factor that shows how appropriate a recommendation would be for this feature value. The needed values for the weights and appropriateness factors have been determined by an online user study (Lerchenmueller 2012).

The third and most interesting component in the context of this paper is the *user activity*. As described earlier, the classification if the user is walking or not is the desired information for the target scenario. In contrast to the other two components, this constitutes a binary value. But as described above, the outcome of each component of the context model is a concrete recommendation score. In order to achieve this, the duration of the user activity is taken into account in case that he / she is walking, leading to the formula $RS_{Activity} = \log(d)$, where d is the duration in minutes if duration > 1 or otherwise d = 1.

Based on the recommendation scores of the three respective components, the global user context score can be calculated by linearly combining the single values allowing for different weights, for example. The resulting recommendation score indicates how appropriate a recommendation at the current time is, based on the current user context. This information can then be used together with other context information in the first phase of the two-phase proactivity model to decide whether to generate a recommendation or not.

Related Work

As this paper focuses on the inference of the user activity as part of the context model, only prior research considering activity classification will be briefly summarized in this section. A recent overview can be found in (Ye, Dobson and McKeefer 2012).

(Kwapisz, Weiss and Moore 2011) use the accelerometer of Android smartphones to collect data about the user activity and infer the type of the activity afterwards. Similar approaches have been taken before, for instance in (Parkka et. al. 2006) where multiple dedicated accelerometers where attached to test persons. Although the accelerometer allows for pretty good activity classification, it can be hardly used in an online application as the update rate is very high and usually pre-processing of the data is necessary. Thus, the global positioning system (GPS), which typically does not require complex pre-processing, will be used for the algorithm in this paper. (Zheng et. al. 2010) presented an approach that only relies on collected GPS data to distinguish various types of user activity. In a first step, a recorded GPS track is divided into

walking and non-walking points based on upper thresholds for the velocity and acceleration. Next, the points are grouped into walking and non-walking segments. Afterwards, various data mining techniques are applied to the non-walking segments to further refine the activity classification.

Inferring User Activity

The related approaches briefly mentioned in the last section, have one important property in common: the user activity is inferred from the collected data offline on an ordinary computer and not on the mobile device. That allows using complex and computationally intense data mining techniques. In our scenario, the algorithm has to run online, i.e. provide results in real-time, on smartphones with limited resources. One important advantage of this approach is that the collected user data remains on the mobile device under the control of the user, which improves privacy. Nevertheless, (Zheng et. al. 2010) provides an interesting approach as they only relied on GPS data. The following description of our developed algorithm will show that their work partially served as a foundation for our approach.

General Idea

As user activity is one part of the user context, its inference follows the same principles as context inference in general. First, raw data needs to be captured by sensors, which can be real hardware sensors but also software sensors. Such information is referred to as the low-level context. In order to make use of the context, usually a more abstract view is needed: high-level context. Considering the user activity, the received raw GPS position can be seen as the low-level context. Knowing that the user is "walking" constitutes the high-level context. In order to infer the high-level context, the captured raw data typically needs to be pre-processed. As already mentioned, data delivered by the device's positioning system can be used without a lot of pre-processing. In how far this step was necessary in our approach will be discussed in the next section. After the pre-processing, the data needs to be interpreted to infer the high-level context. This typically involves various data mining techniques. Furthermore, information from different sources can be combined. After having inferred the high-level context, the final step is to use or apply the context information. In the target scenario, the overall user context score is used to decide whether a recommendation is appropriate or not, as explained above. The user activity is one part of the information that forms the high-level user context. As all steps need to be performed on a smartphone the mobile device, the limited resources of the mobile device have to be taken into account. This relates to computational power as well as to battery life.

Like all current mobile operating system, Android provides access to the device's positioning system via an API. In addition to the GPS, the determination of the position can also be done via WiFi or cell tower triangulation. Although these methods are usually less accurate than GPS, they might be very helpful in areas where the availability of GPS is not guaranteed, such as inside building or parts of big cities. Therefore, the entire device's positioning capabilities are used to achieve best results for our application.

Activity Classification Algorithm

As motivated above, the goal of the activity classification algorithm is to find out whether the user us walking or not. In order to classify the current user activity, the algorithm needs to receive updates about the user's position in terms of a point described by latitude and longitude on a regular basis. Starting with the second point, the following steps are executed.

1. Filtering

GPS data typically does not require pre-processing. However, in contrast to (Zheng et. al. 2010) we apply this step before doing the actual classification. To put it more precisely, filtering is performed to remove invalid points. Such outliers can result from inexact GPS measurements. Filtering out these points has two advantages: computational effort can be saved and the classification described in step 2 is more accurate.

The process of filtering includes the extraction of features. First, a point's accuracy, which is delivered by the positioning system, is considered. If it lies above a certain threshold, the point is filtered out and the next one is examined. Otherwise, the calculation of features is started: distance to the previous point, time difference to previous point, velocity and acceleration. After the calculation of each feature value, the algorithm executes a filter. For example, if the distance between two consecutive points is zero, the point is discarded. Otherwise, the velocity is calculated. If the resulting value lies above a certain threshold the point is again filtered out etc. Thresholds for the filtering were heuristically assigned after some initial tests, while other values were adopted from the work of (Zheng et al. 2010). As mentioned above, such invalid points can occur due to measurement errors and result in unrealistic high values for the velocity.

In addition to these rather simple filtering approaches, the point's (P_i) neighborhood is taken into account, i.e. the previous point (P_{i-1}) and the next point (P_{i+1}). A simple heuristic is then applied to filter out points that follow the

typical outlier pattern of GPS points (Figure 1): if the distance between P_{i-1} and P_{i+1} is smaller than the distance between P_i and P_{i+1} the point P_i is considered as an outlier. The consideration of the point's neighborhood introduces the need for knowing the next point. Therefore, not only the previous point must be remembered, but also the point in question needs to be saved and can not be examined, before the next point is available. Consequently, the algorithm works with a delay of one point and is thus dubbed quasi real-time.

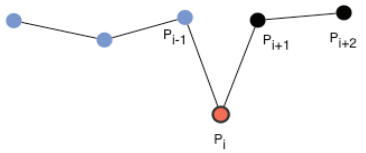

Figure 1: Typical GPS outlier pattern

2. Point-based Classification

When a point is considered as valid, i.e. it is not filtered out, the point-based classification is executed. This is similar to (Zheng et. al. 2010), but is executed in real-time in our case. If the point's velocity exceeds a threshold WT_v or it's acceleration is above a value WT_a, the point is considered as a "non-walking" point. Otherwise it will be seen as "walking" point. At first sight, such a classification might seem sufficient. However, there are some aspects that must not be neglected. If you look at each point separately and only base the classification upon the values for velocity and acceleration, a frequent alternation might occur. Even if the user is walking, points with values that lie above the thresholds can be delivered by the positioning system due to inexact measurements. In our use case, such an alternation is very impractical, as a certain continuity is needed in order to make a statement about the duration of the current activity.

3. Segment-based Classification

To overcome these drawbacks, we apply a segment-based classification. Our approach is different from the one of (Zheng et. al. 2010) concerning the initial classification and the number of segments. Each segment has two properties: the classification (not classified | walking | non-walking) and the state (certain | uncertain). In our case, at most two segments can exist and only one can have the state "certain". As in (Zheng et. al. 2010), we are interested in segmenting a whole recorded track afterwards, the number of possible segments is arbitrary. Furthermore, the classification "not classified" does not exist in their approach. The following paragraphs will illustrate the idea behind the algorithm.

The initial segment is deemed as "uncertain" and "not classified". Each point received after the point-based classification is included in the segment. This means that the segment keeps track of the covered distance and the number of included "walking" and "non-walking" points. If the distance and the number of included points both exceed certain thresholds, the certainty of the segment is evaluated. In the simplest and most successful approach, this is done by comparing the number of included "walking" points against the "non-walking" ones. The classification of the majority is then taken as the classification for the segment, whose state is changed to certain at the same time then.

Now that a certain segment exists, the classification of a new point is compared to the classification of the segment. If it matches, the point is included into the segment. Otherwise, the segment becomes the *oldCertainSegment* and a new uncertain segment with the point's classification is created. Nevertheless, the global classification of the current activity is still given by the *oldCertainSegment*. If the next point's classification matches the one of the new segment, it is included. This continues until enough points are included in the new segment and a long enough distance has been covered to deem it as certain. In this case, the *oldCertainSegment* is discarded. If in contrast, the new segment is still uncertain and a point arrives, whose classification matches the one of the *oldCertainSegment* and not the new one, the information from the uncertain segment as well as the new point are included in the *oldCertainSegment*. Subsequently, the *oldCertainSegment* becomes the only segment again, as the new one is discarded.

The use of two segments and the fact that certainty is not lost right away when a point with a different classification arrives, helps to overcome the problem of having to deal with frequently changing activity classifications. It introduces exactly the continuity needed for our use case. One resulting possible drawback might be a delay in the recognition of real changes in the activity. If the user changes from "walking" to "cycling", for instance, the algorithm might require some time until the new activity is correctly recognized. To what extent this is a problem will be discussed in the section *Evaluation*.

Logging and Classification Framework

In order to test the algorithm, test data was required. For this purpose, we have developed a framework and an application for logging labeled GPS data (and other information) on the Android system. The application provides a simple user interface for starting and stopping the logging process and selecting the current activity.

For the testing of the algorithm, we developed a program that reads the logged position information and gives the

recorded points sequentially to a classification engine that executes the algorithm described above. The given point-based as well as the segment-based classification are then compared to the ground truth, i.e. the user given label. In addition to the classification, this tool also examines timing aspects to analyze the real-time properties.

Evaluation

This section first explains how test data was acquired and then discusses the results of the evaluation.

Setup

For the acquisition of test data, a group of twelve people of mixed gender was equipped with the Android application described at the end of the previous section. Seven participants did not own an Android smartphone themselves and were provided with test devices. Our system does not need special requirements concerning the hardware. The one requirement was to run at least version 2.3.3 of the Android system to guarantee correct operability. In total, four different types of devices were used. This was a good way of verifying if the logging software, and thus also the access of the positioning system, works as desired on different devices. Equipped with devices and software, participants were asked to track and label their activity for a duration of 10 to 14 days. During this time, the participants were asked to collect as much position data as possible and go for a walk of a minimum duration of 30 minutes at least three times.

Description	Amount
Log Sessions	331
Recorded *Position Points*	75146
Given *Annotations*	523
- Walking	230
- Public Transport	100
- Cycling	84
- No Move	68
- Driving	37
- Jogging	4

Table 1: Logged points and annotations

After the user study has been conducted, the logged information was collected from all the devices. The total number of logged points and given labels (annotations) is shown in Table 1. Since test users were instructed to activate the logging mostly when they were moving around, the amount of "no move" points in Table 1 is rather low.

Based on the collected data, the filtering mechanism was examined first. The results showed that filtering should be applied, as a lot of outliers were present in the recorded data. Manual inspection considering also the resulting values for velocity and acceleration showed that the filtered data was much more realistic than the raw data. Figure 2 visualizes an excerpt of the data with some outliers.

Figure 2: Data example

Results

In order to evaluate the results, we were looking at precision of the activity classification algorithm and also timing results. Precision refers to number of matches between the classification inferred by the algorithm and the classification given by the user, i.e. the ground truth.

In more detail, we have obtained the following results (n = total number of classified points):

- Point-based precision: 86.5% (Precision of the point-based classification algorithm, i.e. the number of correct matches, divided by n.)
- Segment-based precision: 85.6% (Number of correct of matches between the classification of "certain" segments determined by the algorithm and the user label for the current point, divided by n.)
- Segment-based precision ignoring "not classified": 92.3% (This metric ignores the cases where the classification of the current segment is "not classified" and therefore deviates from the user label. This property is important because it denotes the number of points that were needed until the first certain classification was made.)
- Segment-based precision ignoring "not classified" and "after change": 96.0% ("after change" denotes the number of points between a changed user given classification and a segment match, where the algorithm's "certain" segment classification is still the same as before the change and therefore differs from the ground truth. Thus, this number shows how many points were still considered as belonging to *oldCertainSegment.*)

In addition to the precision metrics, timing aspects play an important role. As motivated above, our system and algorithm are intended to be used as on online algorithm, i.e. results should be computed in real-time to be able to immediately assess a situation and calculate the user context score. For this evaluation, two different timing properties are be taken into account: the time to the first segment match and the time it takes the algorithm to adapt to changed activity, i.e. getting to a "certain" segment with a classification that matches the new activity. The median for the first property was 95.0 seconds, i.e. it took about one and a half minute for the algorithm for the initial classification. For the second property, the delay it takes the algorithm to correctly classify the activity after it has changed, the results is slightly higher (106.5 seconds) but still very acceptable.

To summarize, the proposed algorithm showed good performance on the collected data set and is easy to implement as an online algorithm on an Android smartphone. One potential problem is to distinguish between activities that may result in similar feature values. Foe example, distinguishing between "driving" in a traffic jam from "walking". We did not test this in more detail because our main goal was just to identify "walking" as user activity, as motivated by the scenario in the introduction.

Conclusion

Proactive recommender systems generate recommendations for the user without explicit user request. This is conceivable in mobile scenarios, for example a tourist visiting and walking around a city. To determine when to generate a recommendation, we have developed a proactivity model that analyzes the current situation. An important part of this model is the *user context* that can be subdivided into *user status*, *device status* and *user activity*. The work presented in the paper focused on the user activity, i.e. figuring out if the user is currently "walking" as an indication for the appropriateness of a recommendation at this time.

We have also worked on investigating the other components of user context. Thereby, we have conducted an online survey to determine the importance of various available features of user status and device status (Lerchenmueller 2012). Next step is to put the different components together and implement the whole proactivity model in a prototype mobile recommender application, and evaluate the complete approach in a realistic scenario. Thereby, an important question is to investigate the power usage by such a system to make it work in practice.

References

Dey, A.K., Abowd, G.D. and Salber, D. 2001 A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human Computer Interaction*, 16:97-166.

Gallego Vico, D., Woerndl, W. and Bader, R. 2011. A Study on Proactive Delivery of Restaurant Recommendations for Android Smartphones, In *Proc. Workshop Personalization in Mobile Applications*, ACM Recommender Systems Conference, Chicago, IL, USA.

Kwapisz, J.R., Weiss, G.M. and Moore, S.A. 2011. Activity Recognition Using Cell Phone Accelerometers. *SIGKDD Explor. Newsl.*, 12:74–82.

Lerchenmueller, B. 2012. Determining Context for a Proactive Recommender System Based on User Behavior. Master's Thesis, Department of Computer Science, Technische Universitaet Muenchen, Germany.

Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J. and Korhonen, I. 2006. Activity Classification Using Realistic Data From Wearable Sensors. Cornell University Press.

Ricci, F. 2011. Mobile Recommender Systems. *International Journal of Information Technology and Tourism*, 12:205-231.

Strang, T. and Linnhoff-Popien, C. 2004. A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management*, UbiComp 2004 Conference on Ubiquitous Computing, Nottingham, UK.

Woerndl, W., Huebner, J., Bader, R. and Gallego Vico, D. 2011. A Model for Proactivity in Mobile, Context-Aware Recommender Systems. In *Proc. of the Fifth ACM Recommender Systems Conference*, Chicago, IL, USA.

Ye, J., Dobson, S., McKeever, S. 2012. Situation Identification Techniques in Pervasive Computing: A Review. *Pervasive and Mobile Computing*, 8(1): 36-66.

Zheng, Y., Chen, Y., Li, Q., Xie, X. and Ma, W. 2010. Understanding Transportation Modes Based on GPS Data for Web Applications. *ACM Trans. Web*, 4:1:1–1:36.