Extracting Semantic Location from Outdoor Positioning Systems*

Juhong Liu, Ouri Wolfson, Huabei Yin
Department of Computer Science, University of Illinois at Chicago
851 S. Morgan (M/C 152), Chicago, IL 60607, USA
{jliu2, wolfson, hyin}@cs.uic.edu

Abstract

With help of context, computer systems and applications could be more user-friendly, flexible and adaptable. With semantic locations, applications can understand users better or provide helpful services. We propose a method that automatically derives semantic locations from user's trace. Our experimental results show that the proposed method identities up to 96% correct semantic locations.

1. Introduction

Context-aware systems often need *semantic location* information that carries semantics about the location. However, most positioning systems (e.g. GPS receivers, PCS network triangulation, proximity sensors, etc.) only provide *physical locations* with time information. For example a GPS point (122.23, 239.11, 11:20AM) indicates that a user m was at geographic location with coordinates (122.23, 239.11) at time 11:20AM. Typical semantic locations are "home, 1200 W. Taylor St", "office, 851 S. Morgan St", or "grocery store, 1340 S. Canal St".

Usually, physical locations are not useful for most users, whereas semantic locations are critical to determine user's activities. They help to deliver better services. For example, consider a user that is watching a movie in a cinema. With user's current semantic location, the context-aware systems can deliver some reminders, e.g. a reminder of closing the cell phone. In addition, the systems understand more efficiently, e.g. in a computer store "apple" means a brand of computer, while in a grocery store, it is a fruit.

Thus, in this paper, we propose a method to determine the semantic locations of a user given a sequence of his/her physical locations in the form of (x, y, t). At a high level, our method works as follows. First, we use a method provided in [9] to extract some 2D positions with geographic coordinates (x, y) where the user spends some time. And then, we use our *reverse geocording* (i.e., translating a physical location to a street address) method to obtain the street address candidates for each 2D position. Then by learning and predicting from all available sources (such as semantic location history, map, yellow pages, address book, calendar, etc), we determine the semantic location of each 2D position.

Our experiments show that the proposed method generates about 96% correct predictions for the combination of frequently and non-frequently visited semantic locations. And for only non-frequently visited semantic locations, the correctness percentage is about 80%.

Before we outline the rest of this paper, let us emphasize here that our method is applicable to both online and offline applications. In an online application, the semantic location is extracted when the user is visiting a physical location, so some services can be provided in real time. While, in an offline application, the semantic location is extracted after the visit, for example, to construct a log file of a user's semantic locations for the memory aid. Furthermore, our context-aware system is designed to run on devices with outdoor positioning and Internet access enabled, such as a smartphone with GPS and Wi-Fi technologies enabled.

The rest of the paper is organized as follows. Section 2 introduces some definitions. Section 3 presents our algorithm to determine the semantic locations. In Section 4, we present the experiment. Section 5 discusses the related works. And finally, we conclude the paper in Section 6.

2. Data model

Typically, a sequence of the (physical location, time) information of a moving object is represented as a trajectory. As defined in [11], a trajectory Tr of a moving object is a piece-wise linear function from time T to 2D space, Tr: $T \rightarrow (X, Y)$, represented as a sequence of points $(x_1, y_1, t_1), \ldots, (x_n, y_n, t_n)$ $(t_1 < \ldots < t_n)$.

Often, the movement of the user is confined in a road network (such as highways, railways, local streets, etc.) given

*This research is supported by NSF Grants ITR-0326284, CCR-0330342, IIS-0513736, and IIS-0209190.

Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)
0-7695-2526-1/06 $20.00 © 2006 IEEE

by an electronic *map*. A map is a spatial object class represented as a relation, where each tuple corresponds to a block with the following attributes:

Polyline: Each block is a polygonal line segment in 2D representing the shape of the street along this block.

L_f_add: Left side from street number

L_t_add: Left side to street number

R_f_add: Right side from street number

R_t_add: Right side to street number

Name: Street name

Type: St, Ave, Blvd, or Rd

Zipl: Left side Zip code

Zipr: Right side Zip code

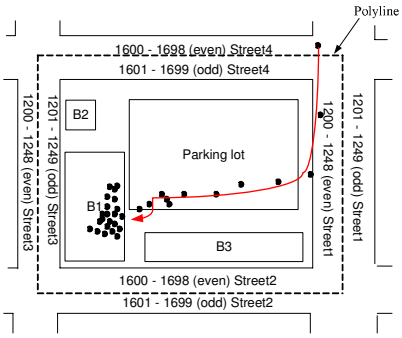

Figure 1. Map, trajectory, and stay

Thus, each street address is on the polyline corresponding to the street block in a map. For example, Figure 1 illustrates four blocks, *Street1*, *Street2*, *Street3* and *Street4* (the four dash lines are the polylines to represent these streets in a map), and a trajectory (in solid black dots) of a user who first driving along street *Street1* then parking his car in the parking lot and finally walking into a building $B1$. Each street has two sides. Each side has either odd or even street numbers, e.g., 1200-1248$(even)Street3$.

Often, the user may spend some time in an area, which is called a *stay* [9]. For example, shopping in a supermarket is a *stay*. As shown in Figure 1, after the user enters the building, he *stay*s there for a while. If the physical locations are sampled at a fixed rate, then they are dense in a time interval. However, driving makes them sparse in the same time interval.

A *semantic location* about (physical location, time) has three attributes, *loc_name*, *semantic_category*, and *street_address*. *loc_name* is the identification of the physical location, for example "BestBuy0312". *semantic_category* represents the semantic category about the physical location, for example "electronic store". *street_address* is the address of the location, for example "1233 S. Halsted St".

The context represents the environment in which a user operates. There are two classes of contexts, *dynamic context*s and static contexts (also called *profile*). The dynamic context includes information pertaining to a transient state in which the user operates. It contains four attributes: *date*, *begin_time*, *end_time*, and *semantic_location*. *semantic_location* is the semantic location of a stay S. *date* is the date in the format of (month/day/year) when S happens. *begin_time* and *end_time* are the time when S starts and ends in that *date*. A sequence of user's dynamic contexts creates a *semantic log file* (or a *log file* for short) of the user. This log file indicates the semantic location history of the user.

The *profile* includes general information of the user pertaining to many states. Typically, the profile of a user contains the following items:

- *Calendar*, recording events in the form of (*event_date*, *semantic_location*).

- *Address Book*, including the semantic locations and phone numbers related to home, office (if available), school (if available), friends or relatives' home, etc.

- *Phone Call List*, recording the semantic locations related to the telephone numbers of the user's outgoing and incoming calls, and the calling dates.

- *Web Page List*, recording the semantic locations about links that were visited on the Internet, and the visiting dates.

- *Destination List*, recording the semantic locations about the destinations whose driving directions are searched on the Internet, and the searching dates.

- *User's Feedback*, indicating whether the predicted semantic locations are correct (*Confirmed List*) or not (*Denied List*).

Usually, *Calendar* and *Address Book* are input by users. They are available on most cell phones, PDAs and Personal Computers. The telephone numbers and date of calls can be automatically retrieved from a cell phone. With a phone number, we search both on the Internet and the Address Book. The semantic location related to a business (if available), with that number, can be retrieved from yellow pages, e.g. switchboard[1]. The semantic location, whose phone number matches the calling number in the Address Book, is also extracted. Thus, the *Phone Call List* can be built automatically. Similarly the *Destination List* and *Web Page List* are built. First, from the web page history in user's web browsers, we get the street address of a business related to the visited web pages, and the street address of the destination. Next, the semantic locations related to the street address can be extracted from yellow pages and the Address Book. For a web page of a business, it is easy to get the contact confirmation: phone number, street address, etc. With these information, *Web Page List* is built. *Phone Call List*, *Web Page List* and *Destination List* are updated every day when the system is idle, i.e. the user doesn't use the system. *User's Feedback* is always input by the user. *Confirmed List* and *Denied List* help in the future prediction.

[1] www.switchboard.com

3. Determination of semantic location

To determine the semantic locations of a user given his/her trajectory, our method works as follows (as shown in Figure 2). First, we scan the (physical location, time) in the user's trajectory one by one, and extract *stay*s with the physical locations (step 1). Then, in step 2, for each extracted stay S, we get all street addresses that are near the physical position of S in a map, using a reverse geocoding method. Thus, for each obtained street address, we search in user's *profile* and yellow pages on the Internet (e.g. switchboard), and get all the *semantic location*s returned by the search as candidates (step 3). Thus, an important problem is to choose one of them as the semantic location of S. We use the following method to make the decision. For each semantic location candidate SL of S, we consider the semantic category of SL, the street address of SL and user's profile as three different information sources. Then, we compute the utility of each source for determining SL, i.e. the similarity between the actual semantic location and SL (step 4.1, 4.2 and 4.3). Next, for SL, we compute the weighted sum of these three utilities, and choose the one with the maximum sum value as the semantic location of S. Finally, the *log file* is updated by adding a new record about this semantic location (step 5).

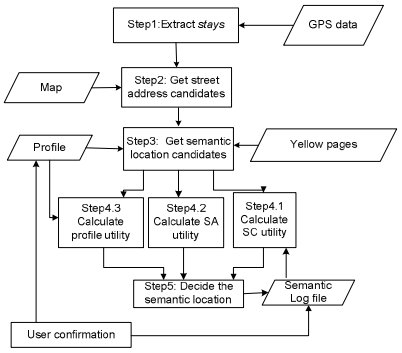

Figure 2. Architecture

Our method requires the participation of users. With user's confirmation, our system will accordingly update the *log file* and the feedback in the profile, which in turn is used for determining the semantic location in the future.

3.1. Extract stays of a trajectory (step1)

In order to extract the stays from a given trajectory for offline applications, we use the similar method introduced in [9, 2]. A stay is that a user spends at least *min_time* in an area where the longest distance between any points in the area is no more than d. For n continuous physical locations $(P_i(x_i, y_i, t_i), \ldots, P_{i+n-1}(x_{i+n-1}, y_{i+n-1}, t_{i+n-1})$, $t_i < \ldots < t_{i+n-1}$, and $t_{i+n-1} - t_i \geq$ *min_time*), a new stay S is generated if in the last *min_time*, the maximum distance of any two physical locations is within d. The center[2] of $(P_i, \ldots, P_{i+n-1})$ is called the physical location of S, denoted as *stay_position*. For a GPS device, it may be the last GPS location that appeared. Because no GPS signal is received in last *min_time* if the user is in the buildings. For a physical location following S, there are two situations. If it is within the distance of $d/2$ miles from *stay_position* of S, it belongs to S, otherwise S ends. A stay has four attributes, *stay_position*, *date*, *stay_start* and *stay_end*, where the last three attributes indicate the date S happens, and the time when S begins and ends respectively.

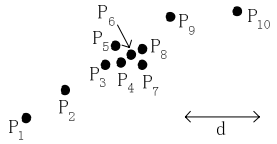

Figure 3. An example for extracting a stay

Figure 3 illustrates an example of the stay generation. In this example we suppose that the positioning system generates a physical location every minute. P_1 to P_{10} represents the 10 physical locations sampled in 10 continuous minutes. Supposing that min_time is set as 5 minutes and d is shown in the figure, the stay is generated as follows. In the first 4 points, no stay is generated, since 4 minutes is less than the min_time. Then, we check the first 5 physical locations, P_1, P_2, ..., and P_5. As the the maximum distance of any 2 points in these 5 points is larger than d, no stay is generated at this time. Similarly for P_2, P_3, ..., P_6. While for P_3, P_4, ..., P_7, a stay is generated, because the maximum distance for these 5 points is from P_3 to P_7, which is less than d. The stay_position is P_4, while the stay_start=3. After that, P_8 is checked. As the distance from P_8 to the stay_position P_4 of current stay is less than $d/2$, P_8 belongs to current stay. When P_9 is checked, the stay ends, because the distance between P_4 and P_9 is larger than $d/2$. So, stay_end=8. Such stay generation process is repeated till reaching the last point of the trajectory.

A similar method is used to extract stays for online applications. But it takes min_time to get a stay after the user stops somewhere.

3.2. Get a set of street address candidates (step2)

The traditional reverse geocoding method, returning the closest street address as the actual street address of a 2D

[2]The center of n points $(P_i, \ldots, P_{i+n-1})$ is P_k $(i \leq k \leq i+n-1)$ such that $\sum_{j \neq k, i \leq j \leq i+n-1} d(P_k, P_j)$ is minimum, where $d(P_k, P_j)$ is the Euclidean distance between two points P_k and P_j.

position, usually generates incorrect results. For example, Figure 4 illustrates a street block in a map, two buildings (B_1 and B_2) and their entrances (E_1 and E_2). The traditional reverse geocoding algorithm returns " 722 W. Taylor St" as the street address of E_2 because E_2 is closest to W. Taylor street. However, actually, the address of Building B_2 is "850 S. Halsted St". In addition, the traditional reverse geocoding often generates an incorrect street address even when the entrance is close to the correct street street enough. For example, building B_1's address is "851 S. Morgan St", whereas, the closest street address is "823 S. Morgan St".

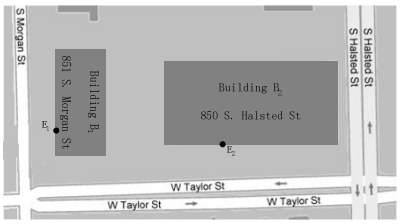

Figure 4. An example for reverse geocoding

To avoid losing the correct addresses, for a given stay S with the stay_position (x, y) in a map Mp, we return a set of street addresses as candidates. The Euclidean distance from (x, y) to the physical location of each street address in this set is less than k. k is a parameter of the system, and is called the *tolerance threshold*.

3.3. Get semantic location candidates (step3)

We get the set of semantic location candidates SLC from the set of street address candidates SAC obtained in step 2, by searching yellow pages on the Internet and the profile of the user. With a street address, yellow pages usually provide the correspondent business name(s), business type(s) (so-called semantic category). Thus for a street address, we get the semantic location triple (loc_name, semantic_category, street_address).

In addition, we extract the semantic locations from the profile, and add them to SLC as candidates. Specifically, given a stay S and one street address candidate SA of SAC, each item in the profile is examined as follows,

- Calendar: for each event e, if $S.date$ is $e.date$ and SA is $e.$*semantic_location.street_address*, $e.$*semantic_location* is returned as one semantic location candidate.

- Address Book: a semantic location in the Address Book is returned as one semantic location candidate, if its street address matches SA.

- Phone Call List: a semantic location in the Phone Call List is returned as one semantic location candidate, if its street address matches SA.

- Web Page List: a semantic location in the Web Page List is returned as one semantic location candidate, if its street address matches SA.

- Destination List: a semantic location in the Destination List is returned as one semantic location candidate, if its street address matches SA.

- User's Feedback: if SA is the street address of one confirmed semantic location $SL_{confirm}$, $SL_{confirm}$ is returned as one semantic location candidate.

3.4. Calculate three utilities (step4)

After step 3, we have a set of semantic location candidates for each stay. Thus we need to determine which one is the true semantic location of a stay S. For each semantic location candidate SL, we consider the semantic category of SL, the street address of SL and user's profile as three different information sources. Then, for determining the likelihood of SL as the true semantic location, we compute the utility of each source in the same scale range [0,100] in steps 4.1, 4.2 and 4.3 respectively.

3.4.1 Semantic category utility (step4.1)

Basically, people visit businesses of different semantic categories according to some rules, e.g. a person shops in grocery stores every Saturday afternoon; one often spends about 40 minutes in a grocery store and two hours in a cinema. For such semantic categories related information is implied in the *log file*. They can help us to predict the semantic location.

We build the *semantic category history* (or *SC history* for short) as follows. For each record in the *log file*, we extract the *start_time* of the stay, $workday\_or\_weekend$, $the\_time\_interval\_spent\_there$[3], and $semantic\_category$ as a new record in SC history. Based on the SC history, we give each semantic location in SLC a *semantic category utility* (or *SC utility* for short). A semantic location with a larger *SC utility* value is more likely to be the true semantic location of the current stay than the one with a smaller value.

We also collect the time information of a stay S, such as the start time stay_start of S, whether S happens in a workday or weekend, and the time interval spent at the 2D position stay_position of S.

Then, we calculate a SC utility as follows. We use the Naive Bayesian model to calculate the probability that the stay belongs to a semantic category C_i appearing in any SL in SLC as:

$$P(C_i \mid T_1, T_2, T_3) = Z * P(C_i) \prod_{j=1}^{3} \{P(T_j \mid C_i)\} \quad (1)$$

[3]This information is obtained by $stay\_end - stay\_start$. It is not available for online applications, so ignore it for online cases.

where T_1, T_2, T_3 are the time-related attributes of the stay: *start_time*, $workday\_or\_weekend$, and $the\_time\_interval\_spent\_there$ respectively. $P(C_i \mid T_1, T_2, T_3)$ is the probability that a stay, with the correspondent values of T_1, T_2, T_3, belongs to C_i. $P(C_i)$ is the unconditional probability of a stay belonging to C_i. $P(T_j \mid C_i)$ is the probability of a record, whose T_j is the correspondent value, in C_i. Z is the value for normalization so that the sum of the all probabilities is 1.

We use the weka software [4] to compute the probability. In order to keep the SC utility in the same scale of the range [0,100] with other two utilities, the value of each SC utility is multiplied by 100.

3.4.2 Street address utility (step4.2)

Now we present the method to compute the *street address utility* (or called *SA utility* for short). The semantic location candidates with different street addresses in SLC have different probabilities to be the correct semantic location of the stay. We set a street address utility for each semantic location candidate SL, based on the street address of SL and the following the observation:

- For the stay_position (x, y) of a stay, the street address of the projection point p on each street has the highest probability thus (i.e. SA utility), to be the street address of (x, y). In addition, the street addresses near p have larger probabilities than the ones far from it.

Thus, if the street address of SL is the street address of the projection point on the same street, then the SA utility of SL is assigned the largest value of the scale, e.g., 100 for the scale of the range [0,100]. Otherwise, the SA utility of SL is inversely proportional to the distance between the correspondent projection point on the same street and the stay_position (x, y). Figure 5 illustrates an example. A denotes the stay_position of a stay S. $A1$, $A2$, $A3$ and $A4$ are the projection points of A on streets $Street1$, $Street2$, $Street3$ and $Street4$ respectively. The street addresses of $A1$, $A2$, $A3$ and $A4$ have the highest SA utility values, while the street address $A1'$ has a smaller SA utility value.

3.4.3 Profile utility (step4.3)

The *profile utility* U_P is calculated from the profile of the user. For each semantic location candidate SL of a stay S, initially, the profile utility U_P of SL has 20 credits. The credits value of U_P of SL increases, if SL matches (or is similar to) the following items of the profile:

- Calendar: SL matches an event e in the Calendar if $S.date$=$e.date$ and SL=$e.semantic\_location$. One match has 80 credits. Not matching has 0 credit.

[4] http://www.cs.waikato.ac.nz/ml/weka/

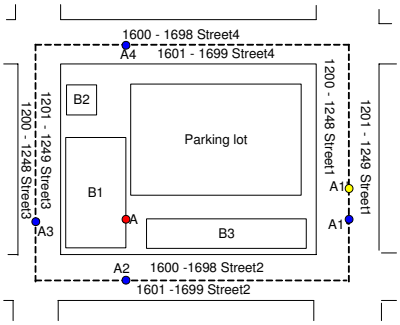

Figure 5. An example for calculating SA utility

- Address Book: If SL matches a semantic location in the Address Book, credit is 70. Not matching has 0 credit.

- Phone Call List: if SL is in the Phone Call List, the closer the call to $S.date$, the higher the credit is; otherwise, the credit is 0. That is, if SL is the semantic location of one call, the credits of the match is (80 minus the number of days between $S.date$ and the date of the call).

- Web Page List: if SL is in the Web Page List, the closer the date when the web page is visited to $S.date$, the higher the credit is; otherwise, the credit is 0. That is, the credits of the match between SL and a web page is (80 minus the number of days between $S.date$ and the date when the web page is visited).

- Destination List: if SL is in the Destination List, the closer the date when the destination is searched on the Internet to $S.date$, the higher the credit is; otherwise, the credit is 0. That is, the credits of the match between SL and a searched destination is (80 minus the number of days between $S.date$ and the date when the destination is searched).

- Confirmed List: SL matches a record in the Confirmed List if SL is found in the list. One match has 70 credits. Not matching has 0 credit.

For each item above, we calculate the matching credits of them[5]. Then we select the maximum value of these matching credits as the increment value of U_P. In addition, if there is a match between SL and one item in the Denied List, the value of U_P decreases by 20 credits. The total decrement is called *debt*. In summary, $U_P = 20 + max(matching\_credits) - debt$.

3.5. Decide the semantic location (step5)

Now, for a stay S, we have many semantic location candidates, each of which has the three utilities. By assigning a weight to each utility, we compute the weighted sum for

[5] The credit can not be negative. Set it as 0 if it is negative

each semantic location candidate SL, as:

$$Sum = w_{SA} * U_{SA} + w_{SC} * U_{SC} + w_P * U_P \quad (2)$$

where U_{SA}, U_{SC} and U_P are utility values for the SA utility, the SC utility and the profile utility respectively; and w_{SA}, w_{SC} and w_P are the weights for each of them. The semantic location candidate with the maximum sum is selected as the semantic location of S.

So the problem is how to set the weight of each utility. Remember that the values of the three utilities are in the same scale. For this situation, in the area of multi-attributes evaluation [4], there are three popular ways to set the weight for each attribute/utility:

- **Equal or unit weighting**: Set all attributes to have equal weights. This method is used when it is hard to give weights to attributes. Or people give widely different weights and it is hard to resolve the conflict.
- **Rank weighting**: Assign the largest rank number to the most important attribute, the next highest number to the second most important attribute, and so on until the least important attribute receives the rank 1. Use the rank as the weight. This method is used when the exact weight is not clear, but people do know the importance sequence of the attributes.
- **Ratio weighting**: Set the ratio for the weights based on experience and/or knowledge. This method is used when people know the importance of attributes very well.

Since we don't exactly know the importance ratio of the three utilities, the third method is not evaluated in this paper. In the next section, we compare the first two methods. For the second method, fortunately, we have only three utilities. Therefore, all the possible ranks can be evaluated, which is 6 in total.

3.6. Initialization

Initially, the *log file* is empty. And user's feedback is not available. In this case, it is hard to decide the semantic locations for user's stays, since the SC utility is not available and the profile utility does not work well without user's feedback. To build the *log file* and user's feedback, we need a stage of initialization. To reduce the disturbance to the user, the initialization process works as follows. The user carries a positioning system (e.g. a GPS receiver) any where he/she goes for collecting the trajectories. After two or three weeks, he/she checks his/her credit card statement, copies and pastes the statement to our system. Basically, the credit card statement shows the information of each purchase: *date*, *business name*, *standard purchase/Internet purchase*, *amount*, etc. For each date, from the trajectory, we extract several stays and then the set of semantic location candidates SLC by the method described in Section 3.1 to Section 3.3. Thus we have two kinds of information coming from independent sources: business names (with date) the user visits from credit card statement, and SLC and time information (date, arrival time, etc.) by our methods. To determine the semantic location for each stay, we just need to match the several businesses in the credit card statement with the SLC by a business name, for every day in the initialization stage. Furthermore, some information in the profile can help to decide the semantic locations. Then the user's confirmation/correction is needed to build the *log file* and user's feedback in the profile.

The user is required to confirm and correct the semantic locations in the initialization stage. After that, only confirmation is required.

4. Experiment

In this section, we first describe our experiment data and settings in subsection 4.1, then present the experimental results in terms of the accuracy of the generated semantic locations in subsection 4.2.

4.1. Experiment data and setting

We have collected the trajectories of a student's trip by a GPS receiver on a PDA for 4 months. The GPS location is sampled every second. The correct semantic location of each stay is recorded manually by the student. So we can compare our predicted semantic location with the correct one. In the 4 months, the student had 224 stays, which belongs to 17 different semantic locations. We transfer the GPS data from the PDA to a PC and do the experiment on this PC.

GPS receivers usually give inaccurate physical locations, especially in the area with trees and buildings nearby. To avoid the noise of the GPS data, we use the *average position* of N seconds[6] to represent the physical position for that time interval. N is a parameter of the system. Our experience shows that 15 seconds is a good value for N. In addition, tolerance threshold k (see Section 3.2) is 0.15 mile.

The profile of this student is built as described in Section 3. There is only one semantic location visit (a hospital visit) recorded in *Calendar*. There is only one link with its semantic location (a government office) is in *Web Page List* and *Destination List*. We do not have any semantic locations in *Phone Call List*. We assume that the user checks the *log file* and gives feedback every week.

[6] The average position of N seconds is a position with 2D coordinates (x, y) such that $x = \frac{x_1+x_2+\ldots+x_m}{m}$, $y = \frac{y_1+y_2+\ldots+y_m}{m}$, $t_1 < t_2 < \ldots < t_m$, and $t_m - t_1$ = N seconds.

We use the map from Geographic Data Technology[7], and online yellow page switchboard in the experiment.

In Section 3.5, we discussed the two methods, equal weighting and rank weighting, for setting the weights of the three utilities. In our experiment, for the equal weighting method, $w_{SA} = w_{SC} = w_P = 1$. For the rank weighting method, we evaluate every possible rank combination for (w_{SA},w_{SC},w_P): (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 2, 1), (3, 1, 2). We randomly select the data of 2 weeks, 3 weeks and 4 weeks as the input of initialization (see Section 3.6). The data for the rest of time is used as the evaluation data. For each setting, the test runs 20 times. In each run, we compute the accuracy of the predicted semantic locations as:

$$Accuracy = 100 * \frac{N_{correct}}{N_{total}} \quad (3)$$

where $N_{correct}$ is the number of correctly predicted semantic locations, and N_{total} is the total number of stays evaluated. We calculate the average accuracy of 20 runs as the final result.

4.2. Experimental result

Our experimental results show that for all evaluated stays, the accuracy of the prediction is about 96% for most (w_{SA},w_{SC},w_P) settings with different initialization data. This is because that more than 76% semantic locations for the stays are the student's office and home. Since these two semantic locations always get high values of the SC utility U_{SC} and the profile utility U_P, we can always predict them correctly. To evaluate the settings of the weights and the initialization data, we remove all the stays for the home and the office. Then test for the other non-frequently visited stays. This test is motivated by the assumption that the user is more interested in the semantic locations he/she visits not very frequently.

The first three columns in Table 1 show the accuracy for non-frequent stays for different settings of the weights and the initialization data.

Intuitively, the initialization data of 3 weeks and 4 weeks get better results than that of 2 weeks. And for the initialization data of 3 weeks and 4 weeks, they get the similar results. This is because that in 3 weeks, our system gets enough information, more initialization data will not improve much. Among different weight settings, (w_{SA},w_{SC},w_P)=(1,2,3) gets the best accuracy. This implies that among all the three utilities, the profile utility is the most important one, then the SC utility is less important, and the SA utility is the least important.

When the user moves to another city, our system can still work without a new initialization. We assume that in the new city, the user keeps his/her visiting habit of semantic categories. So the SC utility helps us to do the prediction. However user's feedback is useless for the new city, since it gives the semantic locations of the original city. Thus, we remove all of user's feedback, and run the tests again to simulate the situations that the user moves to a new city. The last column of Table 1 presents the result with the initialization data of 2 weeks for the new city case.

[7] www.geographic.com

Table 1. Results for non-frequent stays of different initialization data

	original city			new city
w_{SA},w_{SC},w_P	2 weeks	3 weeks	4 weeks	2 weeks
(1,1,1)	82.3	86.1	86.1	77.5
(1,2,3)	89.2	91.9	91.9	79.4
(1,3,2)	78.1	79.7	81.4	76.9
(2,1,3)	86.2	89.4	89.4	72.2
(2,3,1)	76.9	78.1	78.1	76.3
(3,1,2)	78.8	81.9	82.5	68.4
(3,2,1)	76.4	77.4	77.5	76.1

Our system works well with nearly 80% accuracy when the user moves to another city. In this case, the profile utility is still the most important utility. However, it is not so important as that for the original city. Compare the accuracy of weight setting (1,2,3) with that of weight settings (1,1,1) and (1,3,2). From Table 1, we see that for the case of the original city, (1,2,3) improves 6.9% and 11.1% respectively. Whereas for the case of the new city, it improves only 1.9% and 2.5% respectively. This indicates that when the user moves to another city, the importance of the profile utility decreases, and the SC utility becomes more important. This follows the common sense.

All the results shown in Table 1 are collected for the offline trajectories. We also test the online data and get the similar results.

5. Related work

In this section, we compare our work with the literature of context and location information retrieval. Some projects extract user's activity by data from all kinds of sensors. [6] adds acceleration sensors to all major joints on the human body. By collecting and analyzing the data from multiple sensors, they record physical activities of the user: sitting, standing, walking stairs up/down, shaking hands and so on. However, physical activity is not enough to give the information of user's activity, whereas location information gives many clues. For instance, sitting in a theater is watching a movie, while in a meeting room, it is in a meeting.

Easyliving [3] concerns the development of an intelligent environment. It traces the user by location tracing devices

such as active badge systems, cameras, wall switches, and even sensitive floor tiles, etc. Based on the location of the user, it provides he/she the access to information and services, such as home entertainment systems, wall-mounted displays, speakers, lighting, etc. In [5], with a Bat, a user is traced by ultrasonic techniques when he/she moves in a building. With the accurate three-dimensional location, and a semantic map about the building, it is easy to know the location of the user, such as in the meeting room or a laboratory.

However, most of such projects are designed for indoor applications, since it is expensive to install the sensors in a large outdoor area. GPS receiver is one of the few positioning systems which can be used in outdoor environment easily. A typical outdoor GPS application is tour guide, such as Cyberguide [1], which provides the information around users based on the location and the motion direction. Some GPS projects [12, 8] are about user's route. [12] proposes a method to generate the current route of the user, a kind of context, when he/she is driving. [8] uses GPS history data to infer and predict a user's transportation mode, such as walking, driving, or taking a bus. The learned model can predict mode transitions, such as boarding a bus at one location and disembarking at another. Instead of tour guide or user's route, we think the place by which the user stops and stays for a while is important, and the semantic location about this stay is extracted in this paper.

Some works were done to study the stays of users. As most positioning systems only provide physical locations with time information, which is not readable to users, semantic location [10] has to be provided. In some papers (e.g., [10]), semantic location only contains the name of a physical location. However, the name, like "Best Buy", does not give a clue to user's activity, while its semantic category, "an electronics store", is much useful. In this paper, we introduce semantic category, which can be a business type, friend or relative's home, home, office, into semantic location. With semantic category, such as a grocery store or a theater, we can predict user's activity and some services can be provided. In some special cases, when the user moves to another city, our system can predict the new semantic locations by the semantic category history in the original city, while the existing location awareness systems have to learn all from the beginning.

[7] and [2] assume that a stay is to stay in a building and extract that by the fact that GPS loses signals in most buildings. Commotion [7] reminds the user based on his/her current location. It gradually picks up the locations that he/she often visits. After the user gives the to-do list for that location, it gives that list when he/she visits that location again. [2] extracts the significant locations, and then let the user name them. After that, it can predict the movement of the user by a Markov Model. However, sometimes GPS can get signals in a building, based on our experience. And a user may stay at an outdoor place, such as a park. [13] extracts the locations that a user often visits, by a clustering algorithm. [9] extracts the stays and destinations of a user based on different parameters. And it gives models to analyze location histories. All these works require user to give the semantic location for each stay, whereas we obtain semantic location automatically with little help from the user (i.e., user's feedback).

6. Conclusion

In this paper, we argued that the *semantic category*, which can be home, office, friend or relative's home, or business type, was an important context for the user. With *semantic location*, applications can understand users better and provide helpful information to users. A semantic location log file also serves as a good memory aid for users. We proposed a method that automatically derives semantic locations based on the user's trajectory. The experimental results show that our method identifies up to 96% correct semantic locations.

References

[1] G.D. Abowd, C.G. Atkeson, et al., Cyberguide: A Mobile Context-Aware Tour Guide, *Wireless Networks*, 1997.

[2] D. Ashbrook, and T. Starner, Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users, *Personal Ubiquitous Computing*, 2003, (7).

[3] B. Brumitt, B. Meyers, et al., Easyliving: Technologies for Intelligent Environments, *2nd International Symposium on Handheld and Ubiquitours Computing*, Sept. 2000, pp. 12-27.

[4] W. Edwards, and J.R. Newman, Multiattribute Evaluation, Sage Publication, 1982.

[5] A. Harter, A. Hopper, et al., The Anatomy of a Context-Aware Application, *Mobile Computing and Networking*, 1999.

[6] N. Kern, B. Schiele, and A. Schmidt, Multi-Sensor Activity Context Detection for Wearable Computing, *EUSAI*, 2003.

[7] N. Marmasse, and C. Schmandt, Location-Aware Information Delivery with ComMotion, *UHC 2000*.

[8] D. Patterson, L. Liao, D. Fox, and H. Kautz, Inferring High-Level Behavior from Low-Level Sensors, *UBICOMP 2003*.

[9] H. Ramaswamy, and T. Kentaro, Project Lachesis: Parsing and Modeling Location Histories, *GIScience 2004*.

[10] Jorg Roth, The Role of Semantic Locations for Mobile Information Access, *GI Jahrestagung*(2) 2005, pp. 538-542.

[11] G. Trajcevski, O. Wolfson, et al., Geometry of Uncertainty in Moving Objects Databases, *EDBT 2002*.

[12] H. Yin, and O. Wolfson, A Weight-Based Map Matching Algorithm in Moving Objects Databases, *SSDBM 2004*.

[13] C. Zhou, D. Grankowski, et al., Discovering Personal Gazetteers: An Interactive Clustering Approach, *GIS 2004*.