Adaptive On-Device Location Recognition

Kari Laasonen, Mika Raento, and Hannu Toivonen

Basic Research Unit, Helsinki Institute for Information Technology
Department of Computer Science, University of Helsinki
{Kari.Laasonen, Mika.Raento, Hannu.Toivonen}@cs.Helsinki.FI

Abstract. Location-awareness is useful for mobile and pervasive computing. We present a novel adaptive framework for recognizing personally important locations in cellular networks, implementable on a mobile device and usable, e.g., in a presence service. In comparison, most previous work has used service infrastructure for location recognition and the few adaptive frameworks presented have used coordinate-based data. We construct a conceptual framework for the tasks of learning important locations and predicting the next location. We give algorithms for efficient approximation of the ideal concepts, and evaluate them experimentally with real data.

1 Introduction

Location-awareness has several applications in ubiquitous computing. Location-triggered reminders are a simple example (e.g., [1]), adaptive systems that use location as a part of the input for adaptation are more ambitious. Location is also an important clue about a person's communication context, and giving out this information in a form of presence service can enable more efficient communication (see, e.g., [2]).

We present an adaptive framework for building location-awareness from cell-based location data, especially GSM-network cell information. The goal is to be able to learn, within the mobile device, personally important places and routes between them without knowledge of the physical topology of the network.

Most previous work on location-awareness is based on pre-defined location infrastructure and rules about the use of this infrastructure, although the applications do not always strictly demand it. Instead, we are interested in building a learning, adaptive framework for individual location recognition. There has not been very much work in this area, maybe the best-known examples being Marmasse and Schmandt's comMotion [3] and Ashbrook and Starner's work [4]. Both of these approaches use coordinate-based location data, provided by GPS.

Our contributions are twofold. First, we create a conceptual framework for identifying important locations and routes from cellular network data (Sect. 2). The important locations that we can automatically recognize from a user's location data are called *bases*. If the user is moving, we try to recognize the route and aim at predicting the base the user is heading to. Since routes sometimes fork and may lead to several bases, we group bases into *areas* and use these as route targets if a single base cannot be determined with high enough confidence.

Second, we give efficient methods for analysing the observed cellular data on a mobile device, e.g., a cellular phone (Sect. 3). Implementing the adaptive location-awareness system on the mobile device is challenging, mainly because of limited resources. A major advantage is that this enables the building of location-aware systems without additional service infrastructure. Several privacy issues are also avoided, since the location data is held within a device owned by the user.

An experimental evaluation of the concepts and algorithms is given in Sect. 4, using real GSM data from three users, covering a period of six months. Section 5 contains conclusions and outlines future research issues.

2 Locations, Bases, Areas, and Routes

Our goal is to automatically recognize locations that are somehow significant to the user and routes between those locations. As an example application, consider a presence service. We want to describe the current whereabouts of the user. If the user is at a well-known, important place, this is a useful description. If he is not, knowing where he is going to (and coming from) is an informative description of his context. For example, assume that Bob is waiting for Alice at a restaurant. Checking Alice's presence information he could see either that Alice is still at work, has left work and is heading towards the restaurant (or an area including the restaurant) or is heading somewhere else. Based on this information Bob has a fairly good idea of whether he should remind Alice of the meeting or not.

2.1 Locations and Bases

Recognizing locations from GSM cell data is challenging, for a number of reasons:

- Cells can be very large, up to some kilometers in diameter, especially in sparsely populated areas.
- Areas covered by base stations overlap, so that several cells may be seen in a single location.
- Overlap of cells and radio signal shadows can cause cells to be non-contiguous areas.
- There is no one-to-one correspondence between a physical location and the cell used by a phone, e.g., due to changing radio interference.

Other types of cellular networks are likely to have similar properties. An advantage of GSM cellular data is that it is available almost everywhere.

The location data available for a single device consists of a time-stamped sequence of transitions between cells. A correponding cell graph serves as an abstract representation of the cell topology, without reference to any physical locations.

Definition 1 (Cell graph). *The cell transitions of a given user define a directed unweighted cell graph $G_c = (V, E)$, where the set V of vertices consists of all observed GSM cells, and there is an edge $(c_i, c_j) \in E$ if and only if a transition between c_i and c_j has been observed.*

Several overlapping cells can be frequently seen at a single location. In areas with a dense GSM network, it is typical that even if the user stays in one room, the serving cell may oscillate between two or three alternatives. On the other hand, physical back and forth movement most often also indicates a single semantic location, for example when moving around in an office. When such oscillation is observed, we therefore cluster several cells together.

Definition 2 (Cell cluster). *Given a cell graph $G_c = (V, E)$, a set $C \subseteq V$ of cells is a cell cluster, if and only if*

1. *The cells form a subgraph of diameter at most 2 in the cell graph G_c.*
2. *The average length of a visit to the cluster is $t\_avg_C > |C| \max_{c \in C} t\_avg_c$.*
3. *Any proper subset of C does not satisfy condition 2.*

The first condition simply requires that all cells in a cluster are near each other. The second condition tests oscillation: the average time spent visiting a cluster is larger than the sum of the individual times only when the user moves back and forth between the cells in the cluster ($|C| \max_{c \in C} t\_avg_c$ is an upper bound for the sum of individidual averages, and makes the condition relative to the most important cells). Without the last minimality condition some extra cells could possibly be included in cell clusters.

If the user is at a cell that belongs to multiple clusters it is unclear which of the clusters he really is at. There are several alternative ways of dealing with this; for simplicity, we recursively combine all the clusters that have shared cells. This leads to a partitioning of the set of cells to distinct locations. The term "location" is used in this formal meaning in the definitions and algorithms that follow.

Definition 3 (Location). *Given a cell c and a cell graph $G_c = (V, E)$, cell c is at location*

$$\mathrm{loc}(c) = \begin{cases} \{c\}, & \text{if } c \text{ does not belong to any cluster;} \\ \mathrm{cl}(C), & \text{if } c \text{ belongs to cluster } C, \end{cases}$$

where $\mathrm{cl}(C)$ *is the transitive closure of overlapping clusters:*

$$\mathrm{cl}(C) = C \cup \bigcup_{C' : C \cap C' \neq \emptyset} \mathrm{cl}(C').$$

The set of locations is

$$\mathcal{L} = \{\mathrm{loc}(c) \mid c \in V\}.$$

The goal is that locations, as defined above, are the smallest reliably distuingishable units in a GSM cell data. Bases, or important locations, can now be defined simply as locations where the user spends a large portion of his time. However, we want to give more weight for locations visited more recently, in order to adapt to changes in users' movement patterns.

Definition 4 (Bases). *The (weighted) time spent in location L is*

$$\mathrm{time}(L) = \int_{t_0}^{t_{\mathrm{now}}} \mathrm{at}_L(t) r^{t_{\mathrm{now}}-t}\,\mathrm{d}t,$$

where t_0 is the time of the first observation, t_{now} is the current time, $\mathrm{at}_L(t)$ is an indicator function which has value 1 if the user is in location L at time t and 0 otherwise, and $0 < r \leq 1$ determines the rate of aging. We assume time is measured in days.

The set B of bases consists of a minimal set of locations that cover fraction p of all (weighted) time:

$$B = \operatorname*{argmin}_{B' \subseteq \mathcal{L}} \left\{ |B'| : \sum_{L \in B'} \mathrm{time}(L) \geq p \int_{t_0}^{t_{\mathrm{now}}} r^{t_{\mathrm{now}}-t}\,\mathrm{d}t \right\},$$

where $0 < p \leq 1$ defines how large a proportion of the total (weighted) time the bases must have.

In other words, bases are the minimal set of locations in which the user spends proportion p of his total time, taking into account the weighting.

Giving priority to recent events could be accomplished in many ways. Our exponential weight function can be approximated efficiently and incrementally, and its smoothness means that there are no radical changes in the results with advance of time (like there would be with a window with sharp edges).

The aging rate r is determined heuristically. Basically the aging should allow regular events to show their regularity, and not assign overly great weights to daily events. Probably most regular visits happen at least once a week, so a rate that would allow two week old events to have reasonably high weights would be appropriate. Reasonably high weights can be construed as $r^{t_{\mathrm{now}}-t}$ being of order 0.25 at $t = t_{\mathrm{now}} - 14$(days). This gives an estimate $r = 0.9$, which is used throughout this paper.

The proportion p of total time has to be selected as well. We do not present any analytical means here either, but will present some estimates based on our test data in Sect. 4. Too high a proportion will allow purely transitional cells in the set of bases. If too low a proportion is used, not all real significant locations fit in. In our application model, where the user confirms bases by naming them, the former case puts a higher burden on the user but is likely to lead to a more useful set of bases.

2.2 Routes and Areas

The user is not always at a base. Quite often when they are not, they are on their way from one base to another. If we can determine where they are going, this is useful as presence information. For adaptive (user-modelling) applications the from-to base pair can be used as a state to characterize the current context.

The use of cell-based location data again presents some issues:

– We do not know the physical topology of the cells, instead we only know about the times of cell changes.
– Cell changes do not only result from changes in locations, as interference may bump the phone from one cell to another.
– We do not always observe the cell change in the same actual location when moving in different directions. To minimize oscillation (and so network traffic) there is some lag in changing cells to favor the current one.

The first characteristic defines to a large extent what we can infer from the data, and prevents some analytical approaches. The two others make the data fairly stochastic in nature. Our route learning and prediction algorithms are based on sequences of cell transitions.

Routes taken by people may fork or pass by bases (this is typical of public transport as well as main car routes). When travelling along a certain stretch of a route it can be very hard for anyone to predict just from the movement where the person will stop, or which fork they will take. To address a realistic and useful problem, we will group bases into areas of nearby locations, and use these to indicate the approximate direction of movement.

For an area to be a good indication of direction, the bases belonging to it have to be physically close to each other. We do not know the physical (geographical) topology, but we can approximate it with the travel times of the user. These times are used as distance measures between bases, and the bases are then clustered into areas with a density-based clustering algorithm similar to DBSCAN [5]. The base graph represents the base topology.

Definition 5 (Base graph). *The bases define a weighted, undirected graph* $G_b = (B, E, w)$ *where* B *is the set of bases, and where observed transitions between bases define the edges* E*, i.e., if the user leaves base* $b_i \in B$ *and without visiting any other bases arrives at* $b_j \neq b_i$*, then there is an edge* (b_i, b_j) *in* E*.*

The weight $w(b_i, b_j)$ *is given by the median of observed travel times between* b_i *and* b_j *in the graph. The travel time starts when the user leaves the last cell belonging to* b_i (b_j) *and ends when the user enters the first cell belonging to* b_j (b_i)*. The distance is calculated symmetrically using travels in both directions.*

Median is favored because of its robustness in the face of outliers. Especially large travel times that are not representative of the true distance are observed, since the user does not always move continuously from one base to another, but may stop on the way. A minimum would be too vulnerable to a single abnormal transition. If the number of observations grows too large, the median can be calculated approximately [6].

We next define an area of density (travel time) t as a set of bases where any base can be reached from any other base in the area by recursively following edges of weight at most t.

Definition 6 (Area). *Given a base graph $G_b = (B, E, w)$ and a density (travel time) t, the areas (with respect to t) are the connected components of the undirected, unweighted graph $G = (B, E')$ where $E' = \{e \in E \mid w(e) \le t\}$.*

We use a number of densities to obtain a hierarchy of areas. In practice we use a three-level hierarchy, with travel times of $t_1 = 3, t_2 = 10$ and $t_3 = 60$ minutes. These densities are the only parameters to the area clustering. They have been chosen to represent reasonable human perceptions of movement.

Ashbrook and Starner [4] utilize a similar method to group locations, but with only a single layer. These are directly used in their route learning instead of the individual locations. They have a physical topology underlying their location data and have chosen a half mile radius as the threshold, which is (if walking) slightly higher than our first level of $t_1 = 3$ minutes.

We can now formally define the route prediction problem using the concept of areas.

Definition 7 (Route prediction problem). *The input consists of the full history of cell transitions $\{(c_0, t_0), \ldots, (c_n, t_{\text{now}})\}$, and a sequence $\{A_{t_0}, \ldots, A_{t_m}\}$ of partitionings of bases to areas with time densities t_i where $A_{t_0} = B$ ($t_0 = -1$) and $t_i \le t_{i+1}$. Let the previous base visited by the user be b. The task is to output the next base or area the user will visit: the prediction is an area $a \in A_i$ with maximal i such that $b \notin a$, and an estimated probability u for that prediction.*

In other words, the task is to predict the roughest possible area that is different from the previous one. The rationale for this is that making detailed predictions for remote locations is not feasible, but identifying the direction is. In practice, of course, the next base is often in the same area and will be predicted as such. Table 1 illustrates how different kinds of predictions can be interpreted, and how a presence service could describe the user's location to others in various circumstances.

Ashbrook and Starner [4] propose a route recognition model that builds a Markov model of movement between important locations. This is not directly applicable in our work, since we have much fewer bases than they have important locations. The idea of using Markov chains, however, is relevant. The problem of predicting short-term movement in cellular networks has been of interest in systems that use paging. Bhattacharya and Das [7] propose a path learning algorithm for cellular networks based on variable-order Markov chains. We use similar ideas, but do not restrict ourselves to predicting just the next cell but instead try to find the next base.

3 Algorithms

We next consider algorithms to solve the base and route learning problems in a mobile device with limited resources. We assume the availability of 1–2

Table 1. Description of current location in different cases

Analysis	**Example description**
At base	At Work
Not at base, prediction within area with high probability	Left Work 15 minutes ago, heading towards Itäkeskus
Not at base, prediction outside current area with high probability	Left Home 30 minutes ago, heading towards Parents/Jyväskylä center/Summer cottage
Not at base, prediction with low probability	Left Work 10 minutes ago, possibly heading towards Viikki sport center
Not at base, no prediction	Left home 30 minutes ago

megabytes of memory and a peak processing power of 50–100 MIPS. This means that we must store fairly compact statistics of the data and do quite straightforward online processing, and that we are willing to compromise some accuracy in favor of computational efficiency. We also cannot use the available processing power for long periods to avoid draining the battery of the mobile device.

These practical considerations and the requirement of using online algorithms mean that some of the definitions in Sect. 2 cannot be used as is. In particular, algorithms for detecting bases and building locations differ from the corresponding definitions, because the algorithms have to make decisions online, without having access to the full data.

3.1 Bases

The input arrives in cell transition events that contain the new cell identifier c and a timestamp t. The goal is to be able to tell if a cell forms a base or a location, or is part of an existing location. To do this online, we have the following per-location state: $\mathit{time}(L)$ is the total time spent in location L, and $\mathit{count}(L)$ is the number of visits to L. Cell transitions within a single location do not affect the count. The average stay time is defined as $\mathit{avg\_stay}(L) = \mathit{time}(L)/\mathit{count}(L)$.

Let B be the set of bases. To update B when cell transition events occur we run algorithm BaseEvent (Algorithm 1). This routine performs two tasks. First, it updates location statistics (lines 2–4) to determine locations that are visited often or where the user stays for long periods of time. These two factors are combined (line 7) with a weight function that has the form $\mathit{weight}(\mathit{count}, \mathit{time}) = \mathit{time} \cdot \mathit{count}^2/(\mathit{count}^2 + 1)$. The dependence on count is significant only when the count is small. This helps us ignore cells seen only a few times, and has proved necessary in the implementation. Aging according to Definition 4 is approximated outside this algorithm by multiplying each $\mathit{time}(L)$ by the scaling factor r once per day. Finally, the set B is rebuilt on lines 7

```
BaseEvent($c, t$)
Input. A cell identifier $c$ and a timestamp $t$.
State. Set $\mathcal{L}$ of locations, associated statistics, set $B \subset \mathcal{L}$ of bases, previous event
       $\{c', t', L'\}$, base weight threshold $p$.
1: $L \leftarrow$ the location containing $c$
2: $time(L') \leftarrow time(L') + (t - t')$
3: if $L \neq L'$ then
4:     $count(L') \leftarrow count(L') + 1$
5: MergeEvent($L', t - t'$)    ▷ Build locations
6: $c' \leftarrow c; t' \leftarrow t; L' \leftarrow L$
7: $total \leftarrow \sum_{x \in \mathcal{L}} weight(count(x), time(x))$
8: $v \leftarrow$ array of nodes $x \in \mathcal{L}$ sorted into descending order by $weight(x)$
9: $B \leftarrow$ first $k$ entries of $v$ such that $\sum_{i=1}^{k} weight(v[i]) \geq p \cdot total$
```

Algorithm 1. Detecting bases from cell transition events.

to 9. Here the number of bases is determined by taking the proportion p of total weight (cf. Definition 4).

The second task of BaseEvent is to merge neighboring locations to create new, larger locations, according to Definition 2. This is done using a greedy approximation algorithm MergeEvent (Algorithm 2). It follows the most recently seen cells and already formed locations. The history m has to have room for enough items to support merging between μ locations. It turns out that in practice a large majority of mergings occurs between two existing locations, and a few occur between three locations. Accordingly, the maximum size μ of the history was set to four. However, this does not mean that the algorithm cannot produce larger locations. Instead of forming them in a single step, most locations are built from a series of pairwise merges.

In most cases, once a location is formed, it is treated as if it were just a large cell. The algorithm keeps several pieces of information about recent locations: *time*(L) is the total time spent in location L while we have tracked it, *max_stay* is the longest single stay in the location, *avg_stay* is defined above, and *count* is the number of times this location has been seen during merge tracking. Whenever we see a new location (lines 7–16), we consider the previous locations for merging. To be merged, we choose the smallest subset of recent locations that fulfills all the required conditions (line 11).

Definitions 2 and 3 give conditions for clustering based on the average and maximum times spent over *all data*. In the online algorithm, we make decisions based on a single data point for the candidate cluster, since we cannot store statistics for all possible groupings. That a single observation fulfills these conditions is a required but not a sufficient condition for the full data. The additional conditions on time spent and number of visits try to ensure that the data point is not significantly unrepresentative of the full (unknown) distribution.

```
MergeEvent$(L, t)$
Input. A location identifier $L$ and a time $t$ stayed in $L$.
State. List $m$ of recent locations, with associated statistics; maximum size $\mu$ of $m$.

 1: if $L = m_1$ then    ▷ Same as the previous location
 2:    Update $time(m_1)$ and $max\_stay(m_1)$
 3: else if $L = m_i$ for some $i > 1$ then    ▷ Recently seen location
 4:    Update $time(m_i)$, $count(m_i)$ and $max\_stay(m_i)$
 5:    Move $m_i$ to the front of the list $m$
 6: else    ▷ Outside the set of tracked locations
 7:    $k \leftarrow 2$; $merged \leftarrow false$
 8:    while $k \leq |m|$ and not $merged$ do
 9:       $s \leftarrow \{m_1, \dots, m_k\}$
10:       $\tau \leftarrow \sum_{i=1}^{k} time(s_i)$
11:       if $\tau \geq |s| \max avg\_stay(s_i)$ and $\tau \geq |s| \max max\_stay(s_i)$
              and $\tau \geq 10\,\text{min}$ and $\min count(s_i) \geq 2$
              and some item in $s$ is already a base
              and graph formed by cells in $s$ has diameter $D \leq 2$ then
12:          Merge cells in $s$ into a location
13:          $merged \leftarrow true$
14:       $k \leftarrow k + 1$
15:    Remove the last entry of $m$, if $|m| = \mu$
16:    Add a new entry with $\{time = max\_stay = t, count = 1\}$ to $m$
```

Algorithm 2. Building locations by merging existing locations.

3.2 Areas

We use a simple form of density clustering to build areas. Recall that only bases are considered for area clustering. When we enter a base, we update the base graph (Definition 5). The vertices of this graph are the bases, and the (undirected) edges are the observed transitions between bases. Each edge carries with it the average transition time, used as the simplest possible approximation for the median.

Area clustering is performed once per day. The basic clustering step takes a weighted graph $G_b = (B, E, w)$ and a density threshold t. We start the search at an arbitrary node $b \in B$. Then we recursively follow all edges $e \in E$ where the transition time $w(e) \leq t$. All the visited nodes are placed in the same area cluster. If any nodes remain, we choose another node and start the search again and build a new cluster. Nodes whose distance to any other node is larger than t become singleton clusters.

3.3 Routes

Route prediction depends only on cell transitions. When transitions occur, we store them in an event history H. This history contains pairs $h_i = (t_i, c_i)$ of time t_i and cell c_i.

When the user arrives at a base b, the history is used to construct new entries in the route prediction database as follows. We take sequences with a window size k from the history. The first sequence would be $s_1 = h_1 \dots h_k$, then comes $s_2 = h_2 \dots h_{k+1}$, and so on. We then store the associations $s_i \to b$ in the database. Later we can retrieve all the associations that correspond to a cell identifier sequence $c_1 \dots c_k$. Associations are stored repeatedly for decreasing values of k until we reach $k = 1$. When searching for a match, we similarly use progressively shorter sequences until a match is found. Typical initial values of k are about 2 or 3. If $k = 1$, there is no history to tell about the direction of movement; if $k > 4$, most of the sequences we observe will be unique, making it difficult to predict future actions.

We will also experiment with the possibility of using time of day to make more specific predictions. For this purpose, we want to utilize the time distributions of the last transition in the stored sequence s_i. To be able to later reconstruct the distribution of times, in addition to base b we store triplets (n, s, q), where n is the number of occurrences of base b, and s is the sum and q the square sum of the event times.

Prediction of the next base is performed by Algorithm 3. The idea is to take a sequence of recent events as a key and find all the bases stored in the database. If a certain event sequence has led to more than one base, we need to choose the base with the largest probability.

```
PredictBase($H$)
Input.  A history $H = (h_1, \dots, h_n)$ of cell transition events.
Output. A pair $(b, u)$ with a base $b$ and its probability $u$, window size $k$.

 1: $t \leftarrow$ current time
 2: for $i \leftarrow k$ downto 1 do
 3:     $r \leftarrow i$ most recent events in $H$
 4:     $A \leftarrow \{$associations $x \to (n, s, q, b)$ where $x = r\}$
            ▷ $s$ is the sum and $q$ the square sum of $n$ time values.
 5:     if $A \neq \emptyset$ then
 6:         $sum \leftarrow w' \leftarrow 0$    ▷ Sum and the best weight $w$ seen yet
 7:         for all $(n, s, q, b) \in A$ do
 8:             $\mu \leftarrow s/n; \sigma \leftarrow \sqrt{q/n - \mu^2}$
 9:             Assuming $T \sim \mathrm{N}(\mu, \sigma)$, let $w \leftarrow n \Pr(t - \alpha \leq T \leq t + \alpha)$
10:             if $w > w'$ then
11:                 $(b', w') \leftarrow (b, w)$
12:             $sum \leftarrow sum + w$
13:         return $(b', w'/sum)$
```

Algorithm 3. Predicting the next base.

There are two factors that influence the probability: the number of times a base has been seen, and the time distribution. A certain base may be more probable in the morning than in the evening, or the user may have different routes in the weekend than during the work week. These aspects are handled by assuming that the event times follow a normal distribution. This is the simplest possible assumption, as we only need to store the triplet (n, s, q) mentioned above to recreate the distribution. However, experimental evaluation shows that the effect of the time distribution is not as large as it might appear at first.

On line 9 we compute the probability of an association, given the current time of day. Using n and the sums s and q we reconstruct the distribution parameters μ and σ. Then we find the probability of the range $[t-\alpha, t+\alpha]$, where $\alpha = 30\,\text{min}$. The day of the week can be handled similarly (omitted here). We obtain two probabilities which are combined with the number of occurrences n. If the time distribution is not used, line 9 can be simplified by setting the weight w equal to n; it is also unnecessary to maintain the sums s and q.

As an alternative to the presented method we could also predict the next cell instead of the next base. This corresponds to finding the stationary distribution of a kth order Markov chain. This model is more expensive to evaluate and has problems with inevitable loops in the transition graph.

Algorithm 3 yields an estimated probability u for the most likely next base. We can distinguish unsure predictions, with small value of u, from more confident ones. Given a threshold value u', we say that the prediction is confident or has high probability if $u \geq u'$. The final output of the prediction, a base or an area, will be determined as in Definition 7 (see Table 1 for examples).

3.4 Limiting Memory Use

We attach a timestamp to each stored data item, be it a location with accumulated time or a cell sequence. This timestamp is set to the current date and time each time the information is updated. If we run out of resources (or reach a predefined memory limit) we start to remove information starting from the oldest items and continue until the desired level of memory usage is achieved. We assume that there is enough memory to keep at least the "normal" day-to-day schedule of the user in memory.

3.5 Possible Improvements

Bhattacharya and Das present an information-theoretical model for selecting the order of Markov chains to use for each path [7]. This way they try to use an optimal amount of information: if there is a long unique chain of cells in a path, that chain is identified. This could be used in our algorithms as well. We do not want to use too long sequences, though, since we are not really interested in matching the total path travelled, only enough of the recent history to give the direction of travel.

Cell transition sequences are fairly stochastic. For example we have observed that a train trip has only sporadically exactly same sequences when the trip is repeated. Since the sequences do not necessarily match exactly, our route matching is not always able to give good results. We should try to find a suitable metric to define best matches between sequences of cells.

4 Experimental Evaluation

Evaluation of the location model is not straightforward. While the concepts of bases and routes seem intuitively attractive and the clustering of cells and bases is justified by the problem setting, the quantitative quality of the resulting bases and areas is difficult to judge. The route learning has at least one possible objective measure: the accuracy of the predictions based on route recognition.

4.1 Data Gathering

We have gathered cell transition data from three volunteers. The data has been collected for six months with software that runs continuously on the mobile phones the persons use normally (both at work and at leisure). Two of the persons (1 and 2) have fairly simple movement patterns that mainly consist of moving between home and work during the week and some weekend trips. The third user moves somewhat more during the week and visits a larger number of locations.

4.2 Locations and Bases

There are two aspects in judging the quality of the clustering of cells into locations: how well the definitions in Sect. 2.1 work and how well the online algorithm approximates them. Both are somewhat subjective since we do not know the "correct" clustering. Because in the data gathering cells have been named according to the perceived locations, some idea of the quality is given by checking whether the cells in a locations have the same name or not. By looking at the data and the resulting locations, we can say that the original definitions give quite good results. The online approximative version (Algorithm 1) is not too far off. Mostly the problem with the online algorithm is with slightly too large locations: very frequently visited locations tend to assimilate some neighbouring cells as well. Less frequently visited locations are cleaner.

Table 2 shows the bases discovered for person 3. The proportion p (of total time spent in the bases) used in these experiments was 0.8, yielding 37 bases. The aging parameter r is fixed at 0.9 for all tests. The quality of the bases found seems fairly good. Both the stable, recurring locations of everyday life (like work, home, leisure, friends and family) are found, as well as more transient locations on trips (like accomodation in different places). Only few of the bases found are unclear. When p is raised to 0.85 the number of bases for this person grows to 86 and contains quite a few more unclear items.

Figure 1.A shows the number of bases found with different values of p. Larger values potentially allow us to recognize the current location of the user more often. If we assume that the user has to name the bases found, the larger the

Table 2. Description of bases found for person 3

Home	Work
Friends' home	Girlfriend's home
Parents	Girlfriend's parents
Shopping center	Vammala town center
Girlfriend's summer cottage	Student association house
Viitasaari accommodation	Family firm office
Summer cottage	Helsinki center west
Vammala town center 2	Helsinki center east
Vammala church (friends' wedding)	Viitasaari town center
Restaurant	Accomodation in Porvoo
HIIT/ARU office	Porvoo town center
Tvärminne conference center	Sister's home
Friend's grandparents	Friend's home
Accomodation in Tartto, Estonia	A student association
Restaurant	Area near Helsinki center (unclear)
Accomodation in Uppsala, Sweden	Restaurant in Uppsala
Restaurant in Stockholm, Sweden	Unclear
Restaurant in Stockholm	Previous work place
Unclear	

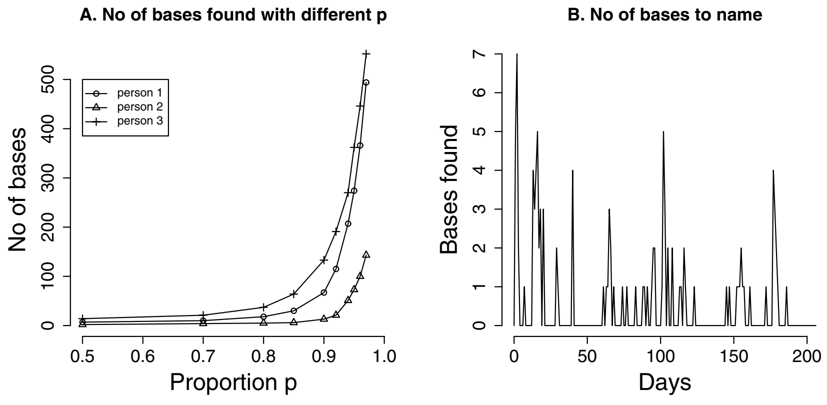

Fig. 1. A: Effect of p on number of bases. B: Number of bases found per day for person 3 with $p = 0.85$

number of bases, the larger the cognitive load on the user. So the chosen p has to balance these facts. Values between 0.8 and 0.9 seem to be reasonable. From the quality of bases found 0.8 would seem best for these users, but 0.85 is not unacceptable, although the data is by no means exhaustive. Adaptively finding a suitable value would be an interesting research topic. When looking at the actual lists of bases, it seems that raising p over 0.85 starts to introduce a large number of locations that do not have a clear meaning to the user in question.

Although the total number of bases the user has to name is important, so is also the peak naming load. Figure 1.B shows the number of bases a user has to name in a day within the data gathering period, with proportion $p = 0.85$. The figure is for person 3, who has the highest number of bases. The naming is distributed fairly evenly over the observation period, although there are some peaks that would probably be annoying to the user.

4.3 Areas

The area clustering seems to work quite well. The first level of the area hierarchy with density $t_1 = 3\,\text{min}$ picks up areas within towns or cities, the second level ($t_2 = 10\,\text{min}$) picks individual cities and the third level ($t_3 = 60\,\text{min}$) regions of countries. For example with the bases of person 3, the first level areas have grouped four districts of Helsinki into their own areas, as well as the town center of Jyväskylä. The second level has grouped the bases in each town in Finland into areas and the third level regions in Finland like Central Finland and the area around Kuopio.

With areas the online algorithms directly implement Definition 6, with the exception that areas are recalculated once a day. The quality of the area clustering depends only on how the approximation of cell clustering affects the selection of bases. The results with online and offline algorithms are almost identical for $p = 0.8$.

The next section shows that the use of areas improves the route recognition results significantly.

4.4 Routes

We evaluate the route prediction by calculating the prediction after each cell transition (unless the user arrives at a base) and by comparing the prediction to the actual base that was reached next. The route learning is done online, so that only data seen up to this point is used in the prediction. We test different variations of the basic algorithms given in section 3 and different parameter settings. The parameters r (aging) and t_1, t_2, t_3 (area densities) have been fixed in these experiments to 0.9 and $(3, 10, 60)$ minutes respectively. In the experiments where p (proportion of total time for bases) or window size are not being varied, values 0.85 and 2 have been used, respectively.

We leave out two cases from the evaluation: when the user is not moving in a well-defined direction and when the next reached base has not been seen before. We say that the user is not moving in a defined direction, if the most recent history of n cells only contains $k < n$ unique cells. For the evaluation we have selected after some experimentation $n = 6$ and $k = 3$. There is no clear evaluation criteria for paths (bases) that have not been seen before, so these are left out.

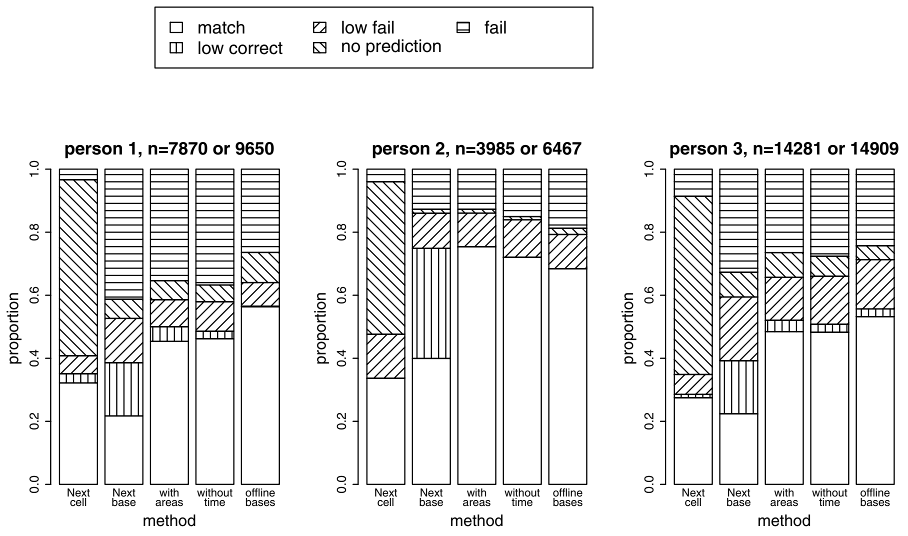

Fig. 2. Route recognition accuracy with different variants of the method. The number of predictions made is n.

Figure 2 shows how well different variations work. The following counts are used in the graphs: *Match* is correctly predicted next base or area with high confidence (the probability u estimated by Algorithm 3 is over 0.3), *low correct* is correct prediction with low confidence. *Low fail* is incorrect prediction with low confidence, *no prediction* means that there was no match for the current sequence of cells, and *fail* is the count of wrong predictions. The methods are: *Without time* is the version that produces the best results overall. It uses Algorithm 3 without the time distribution, and the area clustering for improving accuracy of predictions. The other methods show the effects of varying the algorithm. The first two methods, *Next cell* and *Next base* do not use areas. *Next cell* calculates the probabilites of the next cell and repeats until a base is encountered (using both sequence frequencies and the time distribution) and *Next base* directly calculates the probability of the next base. The remaining methods all use "Next base". *With areas* uses the area clustering (as do the rest). *Without time* is the same except that it uses only the sequence frequencies, and finally *offline bases* presents the results if we calculate both clustering of cells and bases with all the data offline, before running the online route recognition. The impractical offline version helps to evaluate the effect of of the approximations of the online algorithms. The number of predictions n is the same for all the online versions, but lower for the offline variant because the cell clustering is different.

The route prediction results justify the use of the area clustering; it raises the accuracy significantly. It is interesting that using time-of-day affects the results minimally, and not necessarily to the better. This is probably due to assuming a single-mode Gaussian distribution, which is not always true for the data. The last method shown that uses precalculated cell clusters and bases shows that the online approximation is not quite optimal, but it is not too far off either. Calculating next cells instead of bases is not justified, because in addition to being much more expensive computationally, it gives worse prediction results.

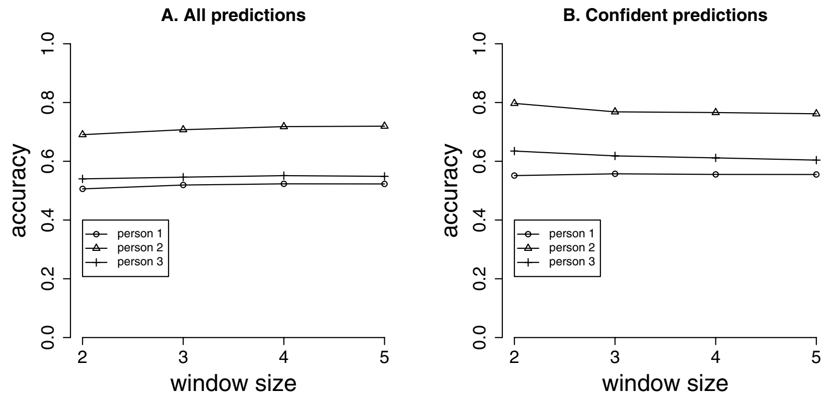

Fig. 3. Route recognition accuracy with varying window size

4.5 Selecting Parameters

Figure 3 shows the effect of the window size k on the route recognition accuracy. The method used is "Without time". Figure 3.A shows the ratio of correct predictions made (both low and high confidence) against all predictions, Fig 3.B the ratio for predictions made with high confidence (estimated probability over 0.3). Lenghtening the window increases the average accuracy of the predictions slightly, but the fact that the longest matches are used results in higher confidences for false predictions as well. Elsewhere in this evaluation we have used window size $k = 2$ as it seems to give the best overall results.

That the prediction accuracy decreases when the window size is increased (especially the for the predictions with high probability) may seem contradictory. The effect is due to overfitting: Algorithm 3 finds the longest possible matching sequence, and assigns probabilities to only sequences of that length. Using all possible matches, but assigning lower weights to shorter matches, could be a useful compromise.

The choice of the proportion p affects both the quality of the bases and the accuracy of the route recognition. Figure 4 shows the effect on prediction accuracy. It seems that to reach prediction accuracy above 0.5, which can be

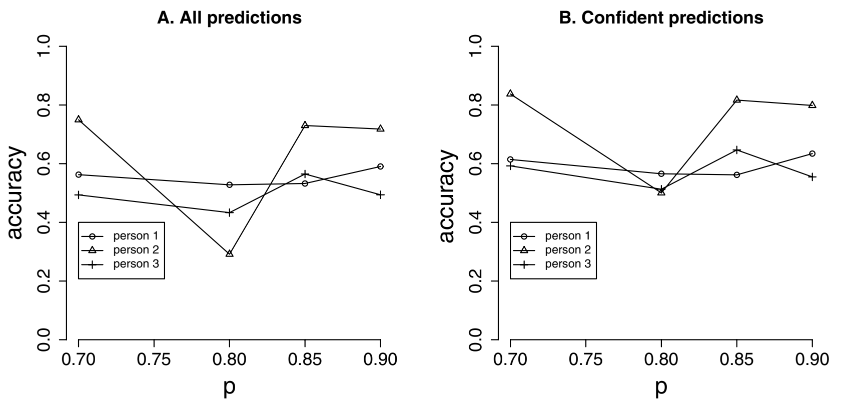

Fig. 4. Route recognition accuracy with varying p

seen as a very rudimentary baseline (that we are correct more often than not), p should be set to 0.85 for our three test persons. However, the accuracy seems to behave in a rather non-monotonic way as a function of p.

5 Conclusions and Future Work

We have presented methods for learning significant locations—*bases*, *areas* of nearby bases, and *routes* between bases from cell transition data. The methods work on a mobile device, reducing privacy issues in the analysis of the data. No new infrastructure or expensive sensors are needed for this kind of location analysis. The results indicate that with some user interaction we can provide interesting presence information from cell-based location data.

The accuracy of the inferred location is limited both by the data and our methods. The impact of this depends on the application. For adaptive applications our framework can be seen as a feature-extraction layer, whose accuracy could be improved using other available data, e.g., from sensors. In presence services, humans seem to be able to augment the location information with other presence data and background knowledge. For example, if the location given by the presence system for Bob is "At home", Alice may infer with fairly high confidence that Bob is on free time at the moment, and either at home or near his home.

The route prediction algorithms presented give adequate results for some applications, but could be improved upon. The algorithms presented use exact substring matching. To improve the recognition accurary we plan to look into string matching techniques that are more suitable for this kind of stochastic data.

Some of the accuracy problems mentioned above have to do with the nature of the GSM cells. Since cells are large, a single cell may contain several significant locations, and often extend to the area surrounding a location. Identifying the individual locations within a single cell cannot be done from the GSM cell data alone. Using the same methods on networks with smaller cell-size, like Wi-Fi and UMTS, would enable us to delineate bases more accurately and add detail to routes. As a preliminary step in this direction we plan to gather data which includes neighbouring GSM cells, so reducing the size of cells to *intersections* of cells.

Probably the most natural alternative to cell-based locationing would be GPS, but it is not without problems either. GPS is not widely available in mobile devices. Further, GPS signal disappears in large parts of urban areas due to the buildings shadowing the signal. Finally, coordinate-based data has the same problem of identifying locations and routes that are useful in personally meaningful location-awareness.

References

1. Dey, A.K., Abowd, G.D.: CybreMinder: A context-aware system for supporting reminders. In: Handheld and Ubiquitous Computing: Second International Symposium, HUC 2000, Bristol, UK, September 2000. Proceedings. Volume 1927 of Lecture Notes in Computer Science., Springer (2000) 172–186
2. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. ACM Transactions on Information Systems (TOIS) **10** (1992) 91–102
3. Marmasse, N., Schmandt, C.: A user-centered location model. Personal and Ubiquitous Computing **6** (2002) 318–321
4. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with GPS. In: International Symposium on Wearable Computing, Seattle, WA (2002)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd international Conference on Knowledge Discovery and Data Mining (KDD 96), AAAI Press (1996)
6. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Random sampling techniques for space efficient online computation of order statistics of large datasets. In: Proceedings of the 1999 ACM SIGMOD international conference on Management of data, ACM Press (1999) 251–262
7. Bhattacharya, A., Das, S.K.: LeZi-update: an information-theoretic approach to track mobile users in pcs networks. In: Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, ACM Press (1999) 1–12