Clustering and Prediction of Mobile User Routes from Cellular Data

Kari Laasonen

Basic Research Unit, Helsinki Institute for Information Technology,
Department of Computer Science, University of Helsinki
Kari.Laasonen@cs.Helsinki.FI

Abstract. Location-awareness and prediction of future locations is an important problem in pervasive and mobile computing. In cellular systems (e.g., GSM) the serving cell is easily available as an indication of the user location, without any additional hardware or network services. With this location data and other context variables we can determine places that are important to the user, such as work and home. We devise online algorithms that learn routes between important locations and predict the next location when the user is moving. We incrementally build clusters of cell sequences to represent physical routes. Predictions are based on destination probabilities derived from these clusters. Other context variables such as the current time can be integrated into the model. We evaluate the model with real location data, and show that it achieves good prediction accuracy with relatively little memory, making the algorithms suitable for online use in mobile environments.

1 Introduction

Location awareness has a large role in ubiquitous computing. Several applications have been proposed that rely on knowing or predicting the location of the user. In this paper we present an algorithm for predicting user movement with respect to cell-based location data. Such location data consists of a sequence of cells, with no regard to physical locations or topology. With this data the task is to learn, on user's personal mobile device, places that are personally important to that user, and to make predictions about the place the user is moving to. Such predictions are useful in, e.g., a presence service, which makes the whereabouts of the user available to other people. Many other proactive applications, such as early-reminder systems [1,2] and traffic planning [3] become possible if we can anticipate the future location of the user.

This paper works with the conceptual model presented in Laasonen *et al.* [4]. The contribution of the present paper is a novel algorithm for predicting routes. The algorithm analyzes whole paths using clustering techniques, instead of relying on the short path fragments of the earlier paper. This both conserves memory and offers better prediction accuracy. The presented approach also respects users' privacy by doing all processing on the mobile phone.

A. Jorge et al. (Eds.): PKDD 2005, LNAI 3721, pp. 569–576, 2005.

Most previous work on determining user locations and routes uses GPS coordinate data [1,2,3]. However, GPS can be problematic in urban areas due to signal shadowing. GPS receivers are also nowhere as ubiquitous as mobile phones. Ashbrook and Starner [1] cluster coordinate data to infer locations, but movement times can be used as well [5]. Alternative methods of prediction of future locations include first or second-order Markov models [1], and Bayes classifiers [2,6].

Our data takes the form of a sequence of cell identifiers. An interesting approach to clustering sequences is with probabilistic suffix trees [7]. Such methods unfortunately require too much memory and processing capacity to be useful in mobile phones.

2 Problem Setting

A GSM phone communicates over the air with a base station. In any given location there may be several base stations whose radio signal reaches the phone. The phone chooses one of them, and switches transparently over to a new base station as needed. A *cell* is the area covered by a single base station; when we say the phone is in some cell, we mean that the phone is in the area of the corresponding base station.

In this paper we work with GSM cell data, for a number of reasons. Mobile phones are ubiquitous and cellular networks are present almost everywhere. Since no operators or external service infrastructure are involved, data gathering is easy and inexpensive. On the other hand, cells may overlap, they vary widely in size, and signal shadowing can make cells appear non-contiguous. Finally, a certain physical location does not have one-to-one correspondence to cells because of radio interference, phone network load and various other issues.

The data consists of *cell transitions*. At the lowest level each cell is represented by an opaque numeric identifier (e.g., "*Sonera.3286.15754*"). Our location data is a time-stamped sequence of such identifiers. We can visualize the data by a graph where the vertices are the observed cells, and there is an edge (c_i, c_j) if (and only if) a transition occurred from cell c_i to c_j. A fragment of such a graph is shown in Fig. 1. This graph shows both the author's daily commute from home ("Vuosaari") to work and trips from home to downtown Helsinki. It does not include transitions in the opposite direction. (For illustrative purposes, some of the cells have been named.)

From our earlier work we will be building on the concepts of cell clusters and bases. If overlapping cells have approximately equal signal strength, the phone may hop between cells even when the user is not moving. This oscillation is handled by *clustering* cells with our earlier method [4]. Intuitively, a cell cluster is a group of nearby cells where most transitions happen within the cluster.

A *location* is either a cell cluster or a single cell. Locations are identifiable in the sense that we can reliably detect the user entering and leaving them. Finally, locations that are important to the user are called *bases*. A location is considered to be a base when the time spent there as a portion of the total time the software

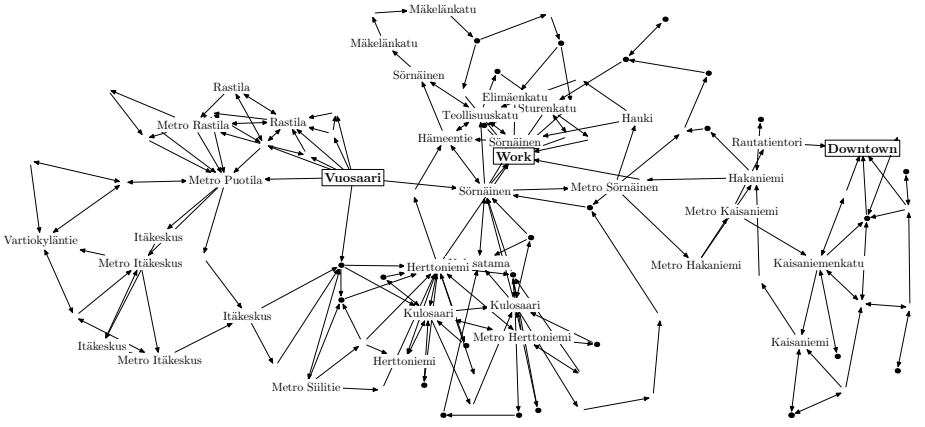

Fig. 1. A partial cell transition graph for routes from "Vuosaari" to either "Work" or "Downtown." Unlabeled dots represent cells that have not been named. The data comes from 69 separate trips.

has been run goes above a certain threshold. The set of bases can change over time, as new places become important or old ones are visited less often. The problem of determining bases is covered in [4]; in this paper we work with a set of known bases. In an actual implementation, learning bases and routes occurs in parallel.

We can now define the problem of *route prediction* as follows: when the user is not in a base, what is the most probable next base? A secondary task is to give some useful characterization of the direction of movement. Furthermore, because the prediction software is run on a mobile phone, there will be tight constraints on the amount of memory and processing power that is available.

Perhaps the most important consequence of using cell-based location data is that we lack the physical topology of the cell network. This includes both the correspondence between cells and physical locations, and all indications of direction. The approach of the present paper is to look at entire routes between two bases, and attempt to learn all different physical routes as strings of cell identifiers. Whenever the user completes a route r between bases a and b, we determine if an existing route between a and b is similar to r. If such a route is found, the two routes are clustered together. Figure 2 shows the effect of applying such route clustering to the data of Fig. 1. There are five different physical routes; the two most frequently traveled are shown in the figure. The graph is obviously much simpler, and furthermore corresponds quite closely to the routes actually traveled in the real world.

Using entire paths makes it also possible to detect *fork points*, which are places where overlapping paths diverge, such as "Sörnäinen" in Fig. 2. When there are several good similarity matches, we can offer a fork prediction as an insurance against the actual base prediction going amiss. From the point of a presence service, a high-confidence prediction of the fork is probably more useful than several low-confidence base predictions.

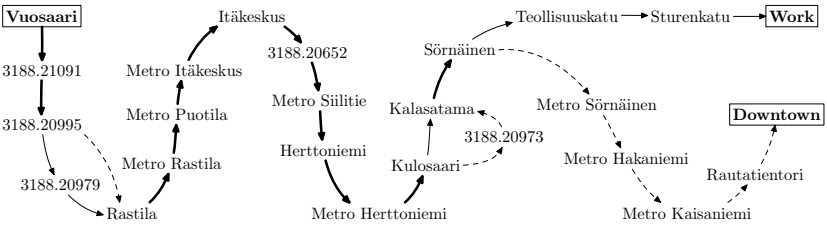

Fig. 2. The most frequent composite routes from "Vuosaari" to "Work" (thin line) or to "Downtown" (dashed line). Edges appearing on both routes are shown with a heavy line. Unnamed cells have numeric identifiers only.

3 Prediction Algorithm

The goal is to predict the next actual base b^*, given that the user's last base was a and since then we have seen a cell sequence $c_1, \ldots, c_k$. When the user is not in any base, at each cell transition we make a prediction, which is a set of pairs (b, p), where b is a possible future base and p the probability of the user going there. When the user arrives at base b^*, the entire route $a, c_1, \ldots, c_n, b^*$ is used to make better subsequent predictions.

3.1 Route Clustering

A *route* is simply a string of cell identifiers. For each pair (a, b) of bases we maintain a set of routes R_{ab}. Instead of storing in R_{ab} all cell paths between a and b, we aim to keep only "typical" paths. Not only does this decrease the memory requirements substantially, but it also proves crucial in estimating the relevance of a given route.

A new route $p = a, c_1, \ldots, c_n, b$ is added to the database when the user arrives at base b, using incremental clustering (Algorithm 1). First p is processed so that only unique cells remain: nearby duplicate cells are collapsed into one. Next (line 3) we determine the similarity of the new path against the existing routes in R_{ab}. If p is similar enough with some route r^*, it is merged with it; otherwise we add p as a new distinct route between a and b.

The similarity function $\mathrm{sim}(r, p)$ tries to approximate the scheme described by Mannila and Moen [8], who use edit distance coupled with item-level similarity. Our version is a heuristic that resembles the Jaccard measure $|r \cap p| \,/\, |r \cup p|$, but enforces ordering for the items. That is, strings r and p are considered equivalent if every element in p appears in r in the same order. Elements in r but not in p are ignored. This asymmetry derives from the fact that a route cluster typically contains more cells than there are in any actual instance of that route. (Algorithm for computing $\mathrm{sim}(r, p)$ is omitted due to space constraints.)

The purpose of merging two paths is to produce a composite path that retains the features of both (similar) participants. We first find the optimal alignment of the two path strings (line 4). Computing the alignment of two strings inserts

ADD-ROUTE(p)
Input: Cell sequence $p = a, c_1, \ldots, c_n, b$, routes R_{ab} between a and b

1 Collapse nearby duplicate cells in p
2 $r^* = \text{argmax}\{\, \text{sim}(r, p) \mid r \in R_{ab} \,\}$
3 **if** $\text{sim}(r^*, p) > \sigma$
4 **then** $r_1, p_1 \leftarrow \textit{align}(r^*, p)$ ▷ Merge p with r^* (see text)
5 $X \leftarrow$ set of letters in $r_1 \cup p_1$
6 **for each** $x \in X$ **do** $v(x) \leftarrow$ average position of x in r_1 and p_1
7 Replace r^* with an ordering of all $x_i \in X$ such that $v(x_i) \le v(x_{i+1})$
8 **else** $R_{ab} \leftarrow R_{ab} \cup \{p\}$ ▷ Add a new distinct route

Algorithm 1. Clustering routes

empty elements ("spaces") into both strings so that identical elements will appear, as much as possible, in the same position [9]. For example, the alignment of "`timers`" and "`tries`" yields "`t␣imers`" and "`tri␣e␣s`". Finally, the merging is completed by ordering all cell identifiers in ascending order by average position in the aligned strings (lines 5–7).

3.2 Making Predictions

Predictions are computed by Algorithm 2, using the previous base a and a history h of m most recently encountered cells. We start by finding S, a set of candidate bases. If $b \in S$, a trip $a \to b$ has been observed. For each b, line 3 computes the similarity of the history h against all possible routes leading to b. A simple prediction system would stop here, and predict that the next base b is the one that maximizes s_b. However, several routes can have nearly equal similarities and still lead to different destinations.

PREDICT-BASE(h, a, A, C, R)
Input: Recent history h, previous base a, context A, context model C, routes R

1 $S = \{\, b \mid R_{ab} \neq \varnothing \,\}$ ▷ Set of candidate bases
2 **for each** $b \in S$
3 **do** $s_b = \max\{\, \text{sim}(r, h) \mid r \in R_{ab} \,\}$
4 Given a and b, find past context data $C_{ab} \in C$
5 Compute $p_b = s_b P(b \mid a, A, C_{ab})$ ▷ See text
6 $b = \text{argmax}_{b \in S}\, p_b$
7 **return** $(b, p_b / \sum_{k \in S} p_k)$ ▷ Return the prediction and its probability

Algorithm 2. Prediction of the next base b

We can choose between destinations by conditioning on additional context variables, such as time of day, weekday and route frequency. We maintain a context database C that stores information from past instances of trips between

pairs of bases. In the most straightforward model we set $C_{ab} = \langle n, T_d(a), T_w(a) \rangle$; this means that for each base pair (a, b) we store n, the number of trips, followed by $T_d(a)$ and $T_w(a)$, distributions of time of day and weekday when the trip started (user left base a). In this case the current context A in algorithm 2 is simply the current time $t = (t_d, t_w)$. We have

$$\begin{aligned} P(b \mid a, t, C_{ab}) &\propto P(b, t \mid a, C_{ab}) = P(t \mid a, b, C_{ab}) P(b \mid a, C_{ab}) \\ &\propto P(t \mid a, b, C_{ab}) \cdot n, \end{aligned}$$

by the definition of conditional probability and the chain rule.

The remaining task is to find the probability of being on the given route at time t. A simple assumption is that the time of day t_d of any given route follows a normal distribution, so we need to store in $T_d(a)$ the sum and the square sum of the previous event times. This allows for later reconstruction of the distribution. For the weekday t_w the normality assumption works less well, so the frequency is used instead. Since t_d and t_w are not really independent, we maintain a separate normal distribution for each weekday, and in the end compute the joint probability as $P(t_w, t_d) = P(t_w)P(t_d|t_w)$.

4 Evaluation

The algorithms were evaluated on the real dataset presented in [4]. The data was collected during six months in 2003 with an early version of the ContextPhone software [10] running on a Nokia 7650 phone. The movements of three volunteers were tracked both at work and at leisure. The movement patterns range from very simple (daily commute, weekend and holiday trips) to moderately complex.

The baseline algorithm is the fragment-based method [4], which was tested with several window sizes k. Since the algorithms are intended for small devices, their memory consumption is also investigated. To simplify the evaluation, both algorithms were tested with offline-given bases, i.e., we did not try to learn both bases and routes at the same time. The algorithms received cell transition events one at a time, supplying a ranked set of predictions for the next base. The top-ranked prediction was then compared to the actual base. Following [4, sect. 4.4], we exclude cases when the user is apparently not moving (stationary).

Figure 3 shows how the different methods compare. Each graph shows how the various prediction algorithms performed. The F_2 and F_4 are the fragment method with a window size of 2 and 4, respectively. The symbol C denotes the route prediction algorithm described in Section 3.2. The model C' additionally includes all intermediate cells and their time distributions. Finally, the bar C_3 shows what happens when the algorithm is allowed to learn each route for the first two times it was seen: prediction results for these instances are not included in the score.

A prediction is *correct* if it matches the actual next base and the probability of the given prediction is larger than $u = 0.3$. A *low correct* prediction is one that is correct, but probability is less than u, or the second-best prediction is

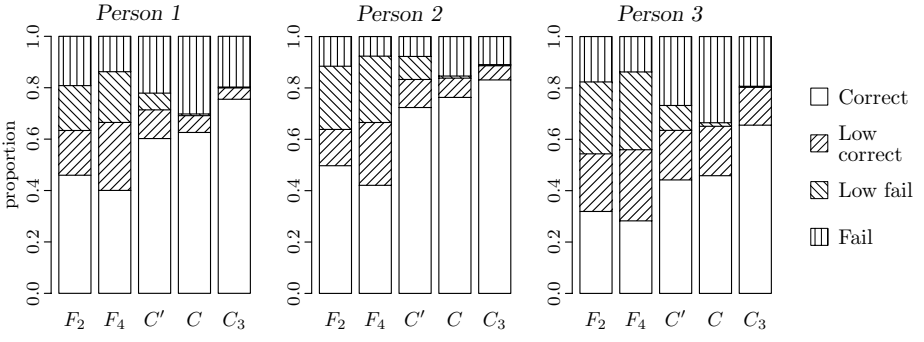

Fig. 3. Route recognition accuracy. Methods F_2 and F_4 are fragment-based. Method C' augments C with intermediate cells; C_3 ignores the first two trips between bases.

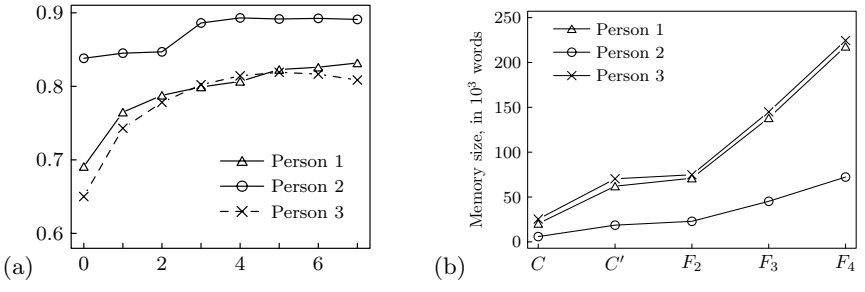

Fig. 4. (a) Prediction accuracy as function of number of learning trips. (b) Comparison of the memory consumption of the algorithms.

correct with nearly equal probability (e.g., $p_1 = 0.55$ and $p_2 = 0.44$), or the fork point was predicted correctly. A *low fail* prediction was wrong, but with a low probability value. Finally, a *fail*-type prediction was a high-confidence prediction that went wrong, or no prediction at all.

For all persons, the route-based method yielded more predictions that succeeded. Taking into account also the low-confidence correct predictions, however, we see more moderate improvements. However, the results so far do not allow for *learning*. As the last (C_3) case of Fig. 3 shows, the prediction quality improves if the algorithm is given time to learn each route. The score is tallied when the pair (a, b) has been seen at least $q = 3$ times. As Fig. 4(a) shows, accuracy improves rapidly with increasing q. The conclusion is that the route-based method is a clear improvement on the fragment method when it comes to prediction accuracy.

Although models C and C' are very similar in their prediction accuracy, the former uses much less memory, as shown in Fig. 4(b). But even model C' consumes less memory than any fragment-based method. For the latter, memory use consists of the fragments themselves and the associated storage for context predictors. The route-based method needs less predictor memory, preferring compact route descriptions.

Because the proposed algorithm is a combination of two separate predictors, it is fairly oblivious to parameter changes. The tests were run with $\sigma = 0.7$ and $m = 12$, which provide a good compromise between quality and efficient use of memory.

5 Conclusion

We have presented a method for predicting user movement from cellular data gathered with user's own mobile phone. The algorithm tackles the problem by attempting to recognize physical routes traveled by the user. The idea is that distinct physical routes correspond to clusters of cell sequences. Later predictions are based on matching the current cell history against known routes. The current time provides additional context to aid prediction. Evaluation of the method with real dataset shows that the method is able to learn and predict routes with good accuracy, while still consuming little memory.

References

1. Ashbrook, D., Starner, T.: Using GPS to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing **7** (2003) 275–286
2. Marmasse, N., Schmandt, C.: A user-centered location model. Personal and Ubiquitous Computing **6** (2002) 318–321
3. Harrington, A., Cahill, V.: Route profiling: putting context to work. In: Proceedings of the 2004 ACM symposium on Applied computing (SAC'04), New York, NY, USA, ACM Press (2004) 1567–1573
4. Laasonen, K., Raento, M., Toivonen, H.: Adaptive on-device location recognition. In: Pervasive Computing: Second International Conference. Volume 3001 of LNCS., Springer Verlag (2004) 287–304
5. Kang, J.H., Welbourne, W., Stewart, B., Borriello, G.: Extracting places from traces of locations. In: WMASH'04: Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, New York, NY, USA, ACM Press (2004) 110–118
6. Patterson, D.J., Liao, L., Fox, D., Kautz, H.: Inferring high-level behavior from low-level sensors. In: UbiComp 2003. Volume 2864 of LNCS., Springer Verlag (2003) 73–89
7. Yang, J., Wang, W.: CLUSEQ: efficient and effective sequence clustering. In: Proceedings of the 19th International Conference on Data Engineering, IEEE Computer Society (2003) 101–112
8. Mannila, H., Moen, P.: Similarity between event types in sequences. In: Data Warehousing and Knowledge Discovery: First International Conference. Volume 1676 of LNCS., Springer Verlag (1999) 271–280
9. Gusfield, D.: Algorithms on strings, trees, and sequences. Cambridge University Press (1997)
10. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: a prototyping platform for context-aware mobile applications. IEEE Pervasive Computing **4** (2005) 51–59