Route Prediction from Mobility Pattern of Cell Phone Users

Tamasree Biswas
Department of Information technology
Narula Institute of Technology, Kolkata, West Bengal, India

Shouvik Halder
Assistant Consultant
Tata Consultancy Services

Mousumi Saha
Department of Computer Science and Engineering
Narula Institute of Technology, Kolkata, West Bengal, India

Abstract- Mobility path of cell phone users plays an important role in context search and advertising, early warning system, traffic planning, route prediction and other applications. In this paper, we present formal definitions to capture the cellphone users' mobility patterns and describes the task of learning routes and predicting future locations by maintaining a database of physical routes. In the pattern discovery phase we use the Apriori algorithm to identify the frequently treaded path. We also use Dempster Shafer Theory of Evidence to predict other locations the user is likely to visit.

Keywords – Parallel Apriori algorithm, Dempster Shafer Theory of Evidence, Route prediction

I. INTRODUCTION

Cellphones have been adopted faster than any other technology in human history and as of 2008 the number of cellphone subscribers exceeds 2.5 billion, which is twice as many as the number of PC users worldwide. Mobility path information of cell phone users plays a central role in wide range of cellphone applications such as context search and advertising, early warning system, traffic planning, route prediction and other applications. The mobility oattern of mobile phone users are here assumed to be captured by the locations of cell towers and the location of cellphone users represented by its nearest cell tower. Path construction is employed to represent the mobility pattern of mobile phone users by an ordered set of cell tower ids. The cell tower ids may also be associated with tags which represent their location name in map. Cell clustering is also employed to prevent dithering between adjacent cells. Thus in its final form entities in the path are cell clusters and adjacent clusters are successive elements in the path sequence.

In the pattern discovery phase we employ the Apriori algorithm to identify the frequently treaded path to make routing predictions for a class of users (e.g. sales team of a company) we are interested in. Route prediction from cellular data does not have the built in advantage of frequent item set mining of Apriori algorithm. Also since we are looking at a class of users, the size of the number of paths on which to mine increases. In order to address the scale of this problem we employ a parallel item set mining algorithm (APRIORI).

For a user who has already visited a history of locations, we apply the Dempster Shafer Theory of Evidence, which gives prediction of other locations the user is likely to visit associated with belief in evidence, this is obtained by application of Dempster Shafer Theory of Evidence to the mined apriori sequences represented as rules. Every sequence mined represented as a rule is associated with a support, confidence and predictive support measure. The basic probability assignment measure associated with consequent places visited is computed using confidence, predictive support measures derived from mined path sequences. The bpa measure is used for final belief computation from presence or absence of consequent place(s) visited.

In section II we will be discussing about the different applications of data mining in various fields. The reason behind this is to give the readers a thorough knowledge about the works that have already been in this field so that

new ideas open up for future. Future scope will be discussed in section III followed by the conclusion in section IV and bibliography in section V.

II. PROBLEM DESCRIPTION

2.1. Overview of Mobility Profiler Framework –

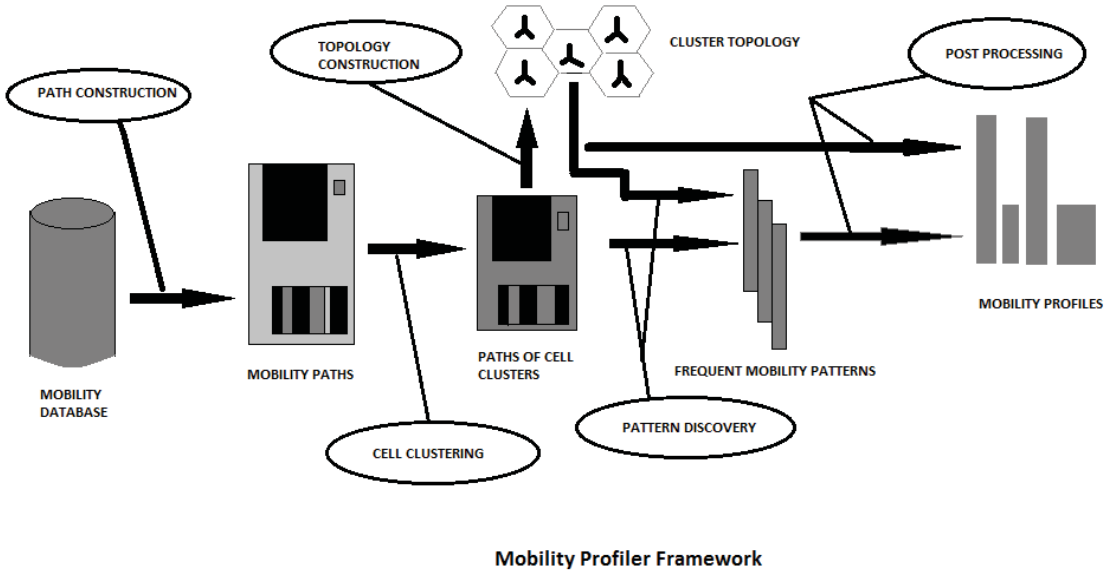

Figure 1 Mobility Profiler Framework

Figure 1 illustrates the general architecture of our framework. We start with 'path construction" to construct ordered set of cell tower ids that correspond to a user's travel from one end location to another. Then, we apply 'cell clustering to gather the oscillating cell towers in the same group and replace the cell towers with the corresponding clusters. After cell clustering, we apply the "topology construction" using the paths of cell clusters as input. In the pattern discovery phase, we discover the frequent mobility patterns of each user separately.

2.1.1 Path Construction Phase

The connectivity information stored in the cellspan table is gathered as follows:
When a cell tower switching occurs, the end time for the previous cell tower is captured and a new record is created in the cellphone that contains the start and end time for that previous cell tower. Simultaneously, the start time for the new cell tower is recorded and is kept until the next cell tower switching occurs. There may also be an unaccounted time gap between two cell towers switching due to disconnection from all base stations or turning off the cellphone.

2.1.2 Cell Clustering

A major problem with the cellular network connectivity data is that a cellphone may dither between multiple cells even when the user is not mobile. A similar problem is also addressed in the Wi-Fi networks referred as the ping-pong effect. The approach mentioned here proactively asserts a simple hexagonal tiling of the cells to restrict the oscillation patterns between atmost three immediate cell neighbor of a point. However, in cellular networks, the geometry connot constraint to be hexagonal tilings, and more significantly most oscillations are due to load balancing purposes and involves arbitrary number of neighbouring cell towers. Therefore, we have proposed two phased reactive approach (decide oscillation after mining real data) to solve this problem.

2.1.3 Topology Construction

Topology construction is used to eliminate majority of candidate path sequences during the pattern discovery phase. By employing the cell cluster neighborhood topology during pattern discovery, the candidate sequences which cannot possibly correspond to a path on the cell cluster topology graph cen be eliminated without calculating their supports.

2.1.4 Pattern Discovery

In this phase, frequent mobility patterns are discovered from mobility paths. We use Apriori algorithm technique for this purpose. This technique is suitable for our problem since we can make it very efficient by pruning most of the candidate sequences generated at each iteration step of the algorithm using the topological constraint: for every subsequent pair of cell clusters in a sequence, the former one must be neighbor to the latter one in the cell-cluster topology graph. An important criteria in our domain is that a string matching constraint should be satisfied between two sequences in order to have support relation. For example, the sequence <1, 2, 3> does not support <1,3> although 3 comes after 1 in both the cases. However, the sequence <1,3, 2> supports <1,3>. A path S supports a pattern P if and only if P is a subsequence of S not violating the string matching constraint. We can apply the paths supporting a pattern as its support set.

2.2 Locations and Bases

A GSM phone communicates over the air with a base station. In any given location there may be several base stations. 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. Cells vary widely in size, and signal shadowing can make cells appear noncontiguous. A software records each cell transition. We can visualize the data by making a graph where the vertices are the observed cells, and there is an edge (c_i:c_j) iff atransition occurred from c_i to c_j.

If overlapping cells have approximately equal signal strength, the phone may hop between cells even when the user is not moving. We handle this oscillation by clustering cells. 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 known as bases. A location is promoted to a base when the time spent there as a portion of the total time the software has been running goes above a certain threshold.

2.3 Route Prediction

There are two approaches for route prediction: The first is to examine the local context of recent cells. Suppose we are in cell c and the previous cells have been $h_1, h_2, h_3, \ldots\ldots$. The idea is to prepare strings $h_k h_{k-1} h_{k-2} \ldots\ldots h_1 c$, for varying values of k. These strings are matched against a database of previously stored fragments. Based on the matches found, and the bases reached from c, we get probabilities for the next base. We begin with, say, k=4. For larger values of k we are wasting storage space, because longer sequences are seen rarely. With smaller k we will get more matches, but at the same time lose the essential context information that we need to establish direction. This method can be augmented with the use of time distribution, which can help us to distinguish between similar routes.

The second approach is more global. Instead of using the local context, we look at entire routes between two bases. We 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 similar to r exists. If such a route is found, the two routes are merged together. To make a prediction, we proceed as follows:

Since we know the user has left base a, we have a set of possible routes and their destinations b. We now use a recent history h of cells and find the route that exhibits the largest similarity to h.

Treating routes as a whole enables a number of features not possible with the fragment method. Firstly, fork points can be detected. A fork point is a place where overlapping paths diverge. When there are several good similarity matches, we can have a fork prediction as an insurance against the actual base prediction going amiss. From the point of a present service, a high confidence prediction of the fork is probably more useful than several low confidence base predictions.

We can also detect backtracking, which happens when the user physically goes some place that is not a base, and comes back. By looking at the entire path, it is not difficult to see if its suffix resembles some earlier subpath, but in reverse. Finally, loop routes, which return to the base from which they started, are rather common.

The problem of route prediction 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 $c_1, c_2, \ldots\ldots , c_k$. There are three phases in the algorithm. First, each time the user leaves a base, i.e. , enters a cell not part of the current base, the system prepares for a new route prediction task. 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. Finally, when the user arrives at a base, the entire route $a,c_1, \ldots\ldots c_n, b^*$ is used to make better subsequent predictions.

2.3.1 Route Composition

For each pair of bases *a* and *b* we maintain a set of routes R_{ab}. When making predictions, we need to match the cell history against the routes R_{ab} for all possible *b*. Instead of storing every encountered path in full, we aim to keep only typical paths. A new route $t=ac_1 \ldots\ldots c_nb$ is added to the database when the user arrives at base *b*. Suppose now that the maximum similarity of *t* against all $r \in R_{ab}$ occurs with some $r = r_{max}$ and is greater than a threshold value σ. Then *t* is merged with route r_{max}. If, however the similarity falls below σ for all existing routes, we add *t* to R_{ab}, the set of routes between *a* and *b*. This process resembles incremental clustering, where each route acts as a cluster, attracting similar routes.

The merging of two paths tries to produce a composite path that retains the features of both participants. We treat the paths as simple strings of cell identifiers, or just letters. First the two strings are aligned by adding empty elements to both. As far as possible, identical letters will appear in the same position in both aligned strings. Computing an alignment of two strings is similar to finding their edit distance, which gives the number of editing operations needed transform one string into the other.

Next we give each letter in both strings a position in the range [0,1]. If the string is $x_1 \ldots\ldots x_n$, the initial value assignment to *i*th letter is simply $v(x_i) = (i-1)/(n-1)$. Now the merged position of letter x is the average position of all nearby occurances of x in both strings. This averaging is only performed within a small window in order to handle cyclic paths correctly. The merged string results from arranging the letters in ascending order by merged position. If two or more letters have the same merged position, their ordering is identified. In subsequent merging letters with undefined positions receive identical position values.

2.3.2 Route Similarity

The similarity function sim(*r;t*) between two routes is used by the prediction algorithm in two cases. In both cases *r* is a composite route between two bases. The other parameter *t* can be a complete path, when the similarity determines the clustering of routes; alternatively, it can be a history of cells (about 10 most recently seen seen cells), and we want to find the route that most closely resembles the history.

2.3.3 Making Predictions

Now we can make the prediction based on the previous base *a* and the already seen cell path $c_1c_2\ldots\ldots c_k$. The entire path is used to detect backtracking; for route matching, a history of most recent is used. The reason not to use the entire path is twofold: we can detect faster that the user is stepping outside any known path, and using shorter strings is in general more efficient.

Route matching has produced a set S of possible reachable bases when starting from base *a*. Making a prediction entails computing for each candidate base $b \in S$ the similarity $\sigma_b = \max \{ \text{sim}(r,h) \mid r \in R_{ab}\}$. In other words, σ_b is the largest similarity of cell history against all routes leading to *b*.

2.4 Parallel Aprior Algorithm

Bodon conducted a literature survey on frequent itemsets mining and presented a fast Apriori implementation focusing on the way candidates generated and the data structures used. The original apriori algorithm was implemented using hash tree. Bodon implemented the Apriori algorithm using the trie structure.

The Apriori algorithm has been revised in several ways. One revision is to partition a transaction database into disjoint partitions. Partitioning a transaction database may improve the performance of frequent itemsets mining by fitting each partition into limited main memory for quick access and allowing incremental generation of frequent itemsets.

Our implementation is a partition based Apriori algorithm that partitions a transaction database into n partitions and then distributes the N partitions to N nodes where each node computes its local candidate k-itemsets from its partition. As each node finishes its local candidate k-itemsets, sends it to local candidate k-itemsets to node 0. Node 0 then computes the sum of all candidate k-itemsets and prunes the candidate k-itemsets to the frequent k-itemsets.

2.5 Dempster-Shafer theory of Evidence

Dempster-Shafer Theory (DST) is a mathematical theory of evidence. In a finite discrete space, Dempster-Shafer theory can be interpreted as a generalization of probability theory where probabilities are assigned to sets as opposed to mutually exclusive singletons. In the DST, evidence can be associated with multiple possible events. As a result, evidence in DST can be meaningful at a higher level of abstraction without having to resort to assumptions about the

events within the evidential set. One of the most important features of Dempster-Shafer theory is that the model is designed to cope with varying levels of precision regarding the information and no further assumptions are needed to represent the information. It also allows for the direct representation of uncertainity of system responses where an imprecise input can be characterized by a set or an interval and the resulting output is a set or an interval. There are three important functions in Dempster-Shafer theory: basic probability assignment function (bpa or m), belief function (Bel), and the Plausibility function (Pl).

The basic probability assignment (bpa) is a primitive of evidence theory. The bpa, represented by m, defines a mapping of the power set to the interval between 0 and 1, where the bpa of the null set is 0 and the summation of the bpa's of all the subsets of the power set is 1. Formally, the description of m can be represented with the following three equations:

$$m: P(X) \rightarrow [0,1]$$

$$m(\phi) = 0$$

$$\sum_{A \in P(X)} m(A) = 1$$

Where P(X) represents the power set of x, ϕ is the null set and a is a set in the power set $A \in P(X)$.

From the basic probability assignment, the upper and lower bounds of an interval can be defined. This interval contains the precise probability of a set of interest and is bounded by two nonadditive continuous measures called Belief and Plausibility.

III. IMPLEMENTATION

The problem that we will be focusing on is to predict the route of the cell phone user from their mobility pattern. Here we will be predicting present location, nearest location and previous five days travel record based on the data provided by the cell tower. We will be using Parallel Apriori algorithm for route determination. Next, our target would be to predict the next location by Dempster Shafer theory. The architecture is shown below:

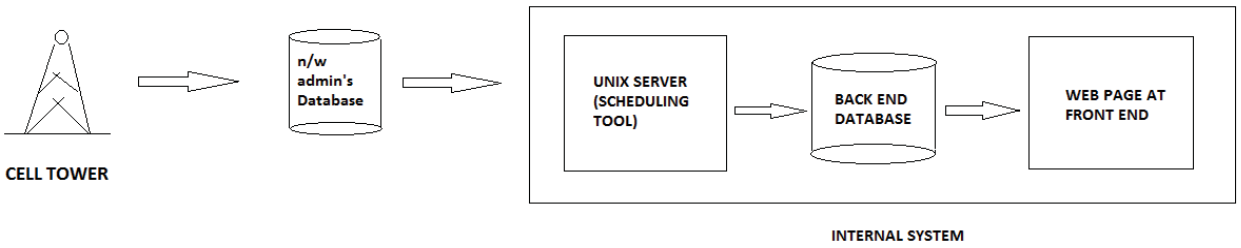

Figure 2 Architecture of the system

Each cell tower, in a location, will contain a cell span table containing the details of the users present in that particular location, at that instance of time. The content of this table will periodically update the database present at the network administrator's end. The network administrator will contain details of the user and the location along with the user_id and location_id.

The internal system will consist of a unix server, the database and the web page from where the user can get necessary information by supplying the user_id and password. The unix server runs a scheduling tool which runs in an infinite mode or scheduled by specific sleep mode to retrieve the data periodically from the network administrator's database. When the data will be retrieved it will populate the database within the internal system.

IV.CONCLUSION

We have proposed a complete framework for discovering mobile user profiles. We have defined the mobility path concept for cellular environments and introduced a novel path construction method. We have also proposed a cell clustering method. The user profiles can be used for city wide sensing application. 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 travelled by the user. Later predictions are based on matching the current cell history against known routes.

Frequent itemsets mining is one of the most important ares of data mining. Existing implementations of the apriori based algorithms focus on the way candidate itemsets generated, the optimization of data structures for storing itemsets, and the implementation details.

Based on the incomplete information about the route of the user we can predict the other locations visited by him. This can be achieved by applying the Dempster-Shafer theory of evidence. The underlying idea is very simple: when presented with a incomplete list of places visited, we first identify all high-support , high-confidence rules that have as antecedent a subsets of s. Then it combines the consequents of all these rules and creates a set of items most likely to complete the route.

As we are able to track the path of individuals, whose records are given to us by the service providers, if all service providers collaborate many criminal activities can be busted as mobile phones are the main linchpin which holds criminal activities and the miscreants together. Another advancement can be done is to manage traffic in specific locality. We can have some threshold value which determines the number of vehicles that can move freely without disrupting the traffic.

REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In ICDE, pages 3–14, 1995.

[2] M. A. Bayir, M. Demirbas, and N. Eagle. Mobility profiler: A framework for discovering mobile user profiles. Technical Report, Deparment of Computer Science and Engineering, University at Buffalo, available at http://www.cse.buffalo.edu/tech-reports/2008-17.pdf, 2008.

[3] A. Bhattacharya and S. K. Das. Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks. Wireless Networks, 8(2-3):121–135, 2002.

[4] A. Harrington and V. Cahill. Route profiling: putting context to work. In SAC, pages 1567–1573, 2004.

[5] M. Gonzalez, C. Hidalgo, and A. Barabasi. Understanding individual human mobility patterns. Nature, 453(7196):779–782, 2008.

[6] N Eagle, A. *MIT Media Lab: Reality Mining*. Retrieved Nov 2009, from http://reality.media.mit.edu/

[7] S. Akoush and A. Sameh " *Mobile User Movement Prediction Using Bayesian Learning for Neural Networks*", Proceedings of the 2007 International Conference on Wireless Communications. (2007)

[8] Liu, T., Bahl, P., and Chlamtac, I, "*Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks"*, IEEE J. Selected Areas Commun., 16 (6), 922-936, 1998

[9] E. Cayirci and I. F. Akyildiz. User mobility pattern scheme for location update and paging in wireless systems. IEEE Trans. Mob. Comput., 1(3):236–247, 2002.

[10] M. A. Bayir, M. Demirbas, and N. Eagle. Mobility profiler: A framework for discovering mobile user profiles. Technical Report, Deparment of Computer Science and Engineering, University at Buffalo, available at http://www.cse.buffalo.edu/tech-reports/2008-17.pdf, 2008.