EECS 545 Final Report

Mobility Pattern Prediction Using Cell-phone Data logs

Hamed Firouzi, Yue Liu and Amir Sadrpour

Abstract:
This paper presents methods to predict the location and dwell time of cell-phone users using a longitudinal study that captures communication, proximity, location, and activity information of the subjects. We have performed three major tasks in this report as follows: the preprocessing in which the raw data was prepared for the next tasks. Location and dwell time prediction using quantized times, and finally location and dwell time prediction using kernel density estimation.

1 Introduction

1.1 Problem Statement

Modeling and understanding people's mobility pattern has attracted more and more interest in this age of ubiquitous communication and computation. Large scale statistical studies have shown that most people have regular daily routines of traveling. This in turn, provides an opportunity for predicting a person's movement given that we have observed this person for long enough time. More specifically, we would like to achieve the following goal in this project: given the observation of a person's mobility history, i.e. a series of locations and corresponding dwell times, predict this person's next location, as well as his/her dwell time in that location.

1.2 Motivation:

Being able to predict people’s movement would benefit a wide range of communication systems. Wireless cellular network can use this information to better allocate resources, for example. In wireless ad-hoc network, more efficient location-based routing protocol can be designed. It also opens new chances for interesting applications and services in today’s pervasive communication and computing environment, such as innovative location-based services. Although a lot of research has been done to predict people’s next location, we have not found dedicated research to also predict the dwell time of next location. Intuitively it requires a prediction algorithm that is able to handle the mixed model of discrete location and continuous time.

1.3 Related Works

A large body of literature has focused on using techniques such as dynamic Bayesian network, neural network and text compression algorithms to do locations prediction [4][5][7][8]. Cheng et al’s survey paper [1] classified these algorithms into two groups: 1. Domain independent algorithms that take results from Markov analysis or text compression algorithms, such as Lempel-Ziv algorithm. 2. Domain-specific algorithms that consider the geometry of user motion as well as the semantics of the symbols in the user’s movement history, such as Liu et al’s hierarchical location prediction scheme[3] for location management in wireless ATM environment. In our work a simple Markov chain model is used and the result for location prediction is satisfactory.

1.4 The Model under Consideration

We consider people's movement as a random process of locations: $X_1, X_2, \ldots, X_n$, where X_i denotes the person's location after the *i*-th location transition happens. It has been demonstrated by previous work that this random process can be modeled as an order-k Markov process. Let's define the person's mobility history as $H_n = \{X_1 = a_1, X_2 = a_2, \ldots, X_n = a_n\}$, and let A denotes the set of all possible locations, then formally we want to find $arg \max_{a_{n+1} \in A} P(X_{n+1} = a_{n+1} \mid H_n)$. Since it is a Markov random process, $P(X_{n+1} = a_{n+1} \mid H_n)$ can be reduced as follows:

$$P(X_{n+1} = a_{n+1} \mid H_n) = P(X_{n+1} = a_{n+1} \mid X_{n-k+1} = a_{n-k+1}, \ldots, X_n = a_n)$$

Given large enough set of H_n, we should be able to approximate the above probability as

$\hat{P}(x_{n+1} = a_{n+1} \mid H_n) = \frac{N(a_{n-k+1} \ldots a_n a_{n+1})}{N(a_{n-k+1} \ldots a_n)}$, where $N(s)$ denotes the number of times sequence s occurs in the mobility history H_n.

1.5 Data Description

We use the data set collected by the MIT Reality Mining project to verify our idea [9]. It is one of the largest mobile phone experiments attempted in academia, including 106 participants carrying around cell phones installed with a logging software, for a duration of nine months. More specifically the software keeps log of the associated cell phone tower ID and the occurring time whenever a hand-off happens. This data set was used for the purpose of our study since a cell phone tower ID is an indication of user's location. However, there are a few complexities with the raw data which require some preprocessing. The first problem is caused by the base station overlap. Another problem is that this data set does not provide information about the physical locations of cell phone tower and we have to figure out a way to approximately define the distances between different towers. More detailed description of these problems are given in 'Preprocessing' part.

2 Preprocessing

2.1 The Need for Preprocessing

The Reality Mining database includes time-stamped tower transitions. The time, the tower ID and the cell ID were recorded whenever a subject's cell phone made a transition from one tower to another. A tower can contain multiple cell IDs. In this context, the tower ID provides a wide estimation for the subject's location, and the cell ID narrows down this estimation. "An important characteristic in any cellular network is that areas covered by base stations overlap, so that several cells may be seen in a single location. If overlapping cells have approximately equal signal strength, the phone may hop between cells even when the user is not moving [5]." Consequently tower transitions are not necessarily an indication of an actual change in the location. Figure 1-(a) shows, the time-stamped rapid tower transitions for a short period of time between tower 30 and 34. To overcome this problem, we cluster cells that tend to represent one physical location using the spectral clustering technique.

2.2 Graph Clustering: Spectral Clustering

The goal of (spectral) clustering is, given a set of data points $X_1, X_2, \ldots, X_n$, to find clusters such that data points in the same cluster are similar and points in different clusters are dissimilar to each other with respect to some similarity/dissimilarity measure. Here, we implemented a method using the spectral decomposition of the similarity matrix based on graph theory. We directly calculated the similarity matrix from our raw data. Based on our definition of similarity, two towers are similar if large number of transitions occurs between them in a short time span. This notion has been elaborated in more details in the “adjacency matrix” section.

The first observation was that the total number of different (unique) tower IDs for all 106 subjects turned out to be around 18,000. We learned that a large portion of the towers were observed/transitioned to only a few times in the entire study. In order to better understand the frequencies, we constructed a Pareto Chart. The Pareto chart is a frequency distribution (or Histogram) of attribute data arranged by category [6]. Figure 1-(b) shows the unsorted histogram of the frequency of tower observations. This graph shows the drastic difference in the frequency of appearance of towers. Certain towers are transitioned into thousands of times, while some others have frequencies under 10.

18,000 unique tower IDs will pose a significant challenge on the clustering. Large number of tower IDs will increase the dimension of the adjacency matrix. As we will show later, this will result in considerable computational burden when finding associated eigenvectors and eigenvalues when performing spectral clustering. We learned that excluding the bottom 10% of the towers that have very low appearance in the data base, will reduce the number of unique towers to around 1,200. These towers appear in the data logs with very high frequencies, and play a significant role in approximation of the movement pattern of subjects.

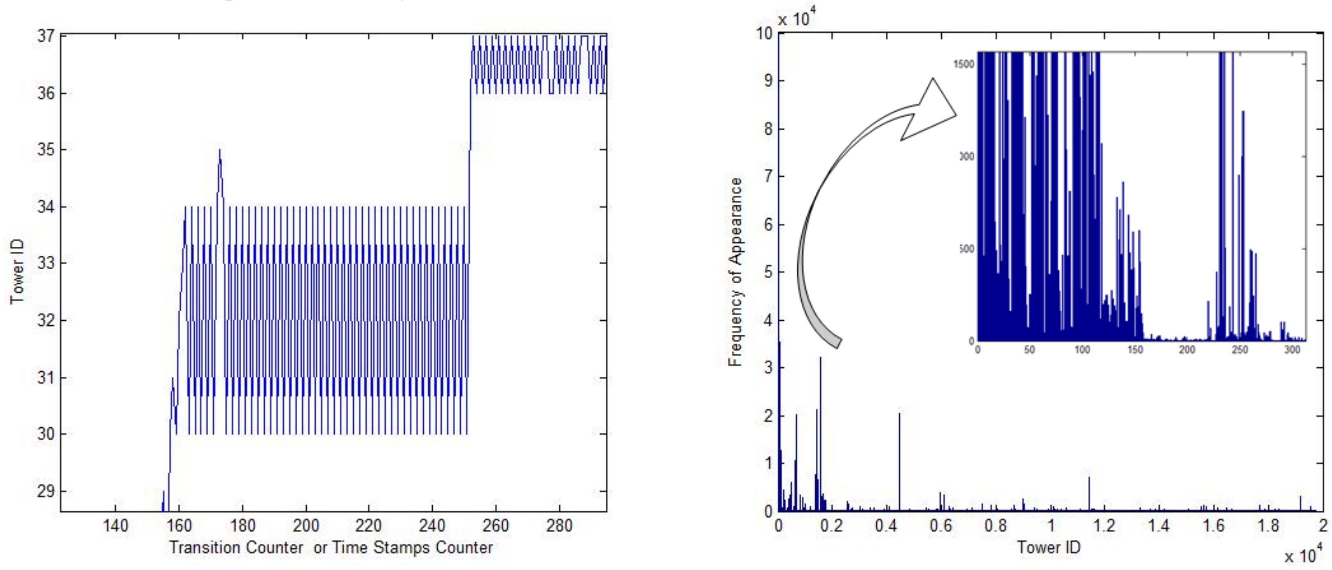

Figure 1: Frequency of appearance of towers. (a) rapid oscillation between towers. (b) The contrast between frequencies of appearances is very large. Some towers only appear a few times.

2.3 Adjacency Matrix

In order to perform clustering, we constructed a weighted graph. The vertices of this graph were towers selected in the previous section. If we denote time of the n^{th} transition by t_n, the vertex of tower i by V_i, and if (V_i, t_n) indicates that we are in vertex i at time t_n, then there is an edge between V_i and V_j if there is a n for which pairs of $(V_i, t_n), (V_j, t_{n+1})$ exist. The weight of the edge is defined by the number of these transitions back and forth between the two towers.

2.4 Spectral Clustering using K-mean Neighborhood

To perform spectral clustering, we followed the procedure as follows [2]:
The summation of each row of the adjacency matrix was calculated and placed in a diagonal matrix.

Laplacian matrix was then constructed by finding the difference of the diagonal matrix above with the adjacency matrix as shown below:

$$\text{AdjacencyMatrix} = \begin{bmatrix} a_{1,1} & & & \\ & a_{2,2} & & \\ & & \cdots & \\ & & & a_{1186,1186} \end{bmatrix} = [A] \quad \text{Then } [A]\begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_{1186} \end{bmatrix} \Rightarrow \mathit{Laplacian} = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \cdots & \\ & & & d_{1186} \end{bmatrix} - [A]$$

In the next step, we calculated the eigenvectors and eigenvalues of the Laplacian matrix. The actual clustering was implemented on eigenvectors corresponding to the smallest set of eigenvalues. To find the right number of eigenvectors, we drew the sorted graph of eigenvalues as shown in the next figure. In particular we are interested in large jumps (elbows) in the graph to set a threshold for the number of eigenvectors that are needed for clustering. Figure 2, shows the first two relatively large jumps at eigenvalues 24 and 45. The 45th eigenvalues is slightly more than 1. This value can be considered too large, and over fitting may become a concern if we use such a large number of eigenvectors for grouping. However, neither of the jumps is large enough to provide strong confidence for selecting a certain threshold. In practice, to obtain training data for mobility pattern prediction, we attempted to use as few numbers of eigenvectors as possible. Therefore, 24 eigenvectors appeared to be more reasonable for generating clusters of towers compared to 45 eigenvectors.

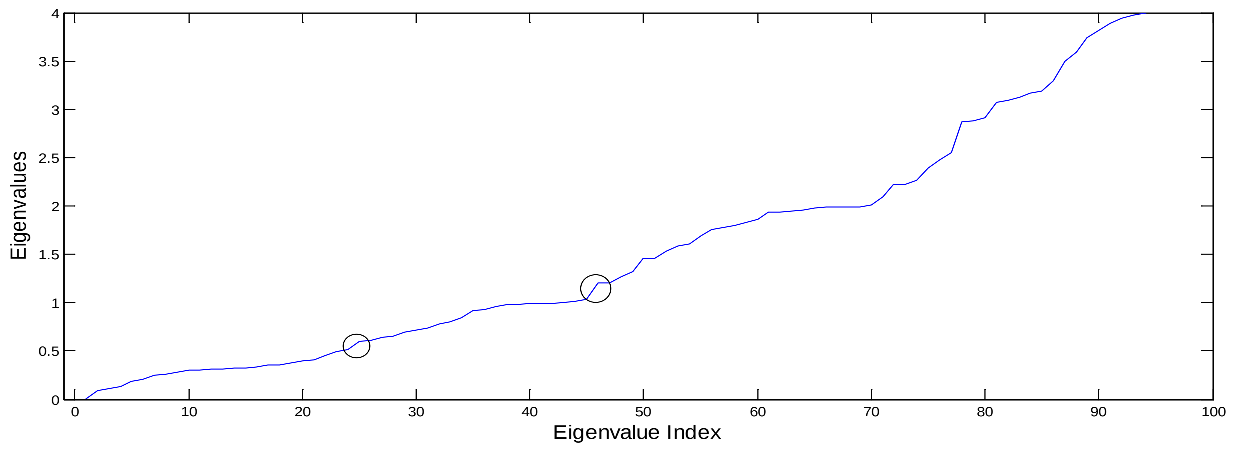

Figure 2: Calibrating Number of Clustering Eigenvectors

2.5 Visualizing Clusters

In order to visualize the spatial map between towers in each cluster, the first two eigenvectors of the Laplacian matrix was used. Since towers correspond with rows eigenvectors, each row of the two eigenvectors can be thought of as two coordinate axis of the associated tower. This method appears to work best when we assume there are two major clusters in the data. This is due to the fact that in the ideal case, where there are K disjoint clusters and K is known, each eigenvector distinguishes one cluster. Additionally, the above visualization method was extended to the three-dimensional space using the first three eigenvectors. The results are shown in Figure 3.

In order to get more detailed movement patterns, we needed more clusters. In other words, more clusters resulted in less towers in each group and the integrity of movements were better preserved. We also noticed that with more eigenvectors (dimensions), it became more and more difficult to obtain larger number of clusters. For instance, with 45 eigenvectors for clustering, we initially could not achieve 60 or more clusters as several clusters would have zero members; whereas, with 3 or 4 eigenvectors even 200 clusters were achievable. This might be due to that fact that in higher dimensional space, it is more difficult to find towers that are similar. This problem was alleviated when we performed a preliminary clustering phase on a random 10% subsample of eigenvectors' rows (corresponding to 10% of towers). This preliminary phase is itself initialized using k-rows (corresponding to k-towers) from eigenvectors at random.

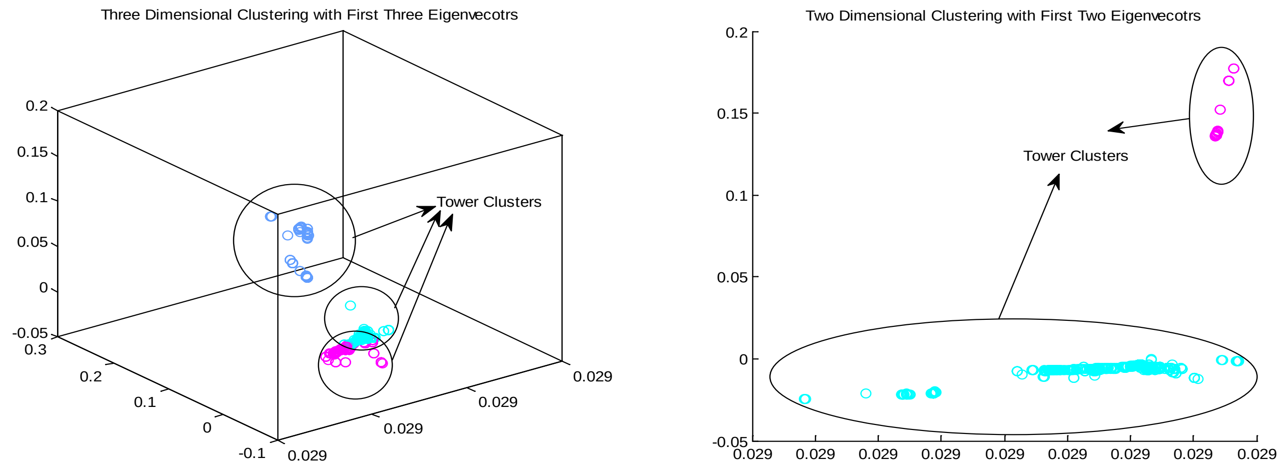

Figure 3: Visualizing Clusters Using the First 2 and 3 eigenvectors. The graphs show K-mean performance in each case

2.6 Mobility Aggregation after Clustering

As mentioned earlier rapid oscillations between towers creates challenges regarding the true location of the subjects when studying their mobility patterns. Spectral clustering removes part of this problem by placing towers with high frequency of oscillations into one cluster. Clustering on the other hand, would aggregate the movement of the subjects as within cluster movements are removed from the model. The diagram below shows the tower transitions and their weights (frequency of transition) before and after clustering for subject 3 in our data set.

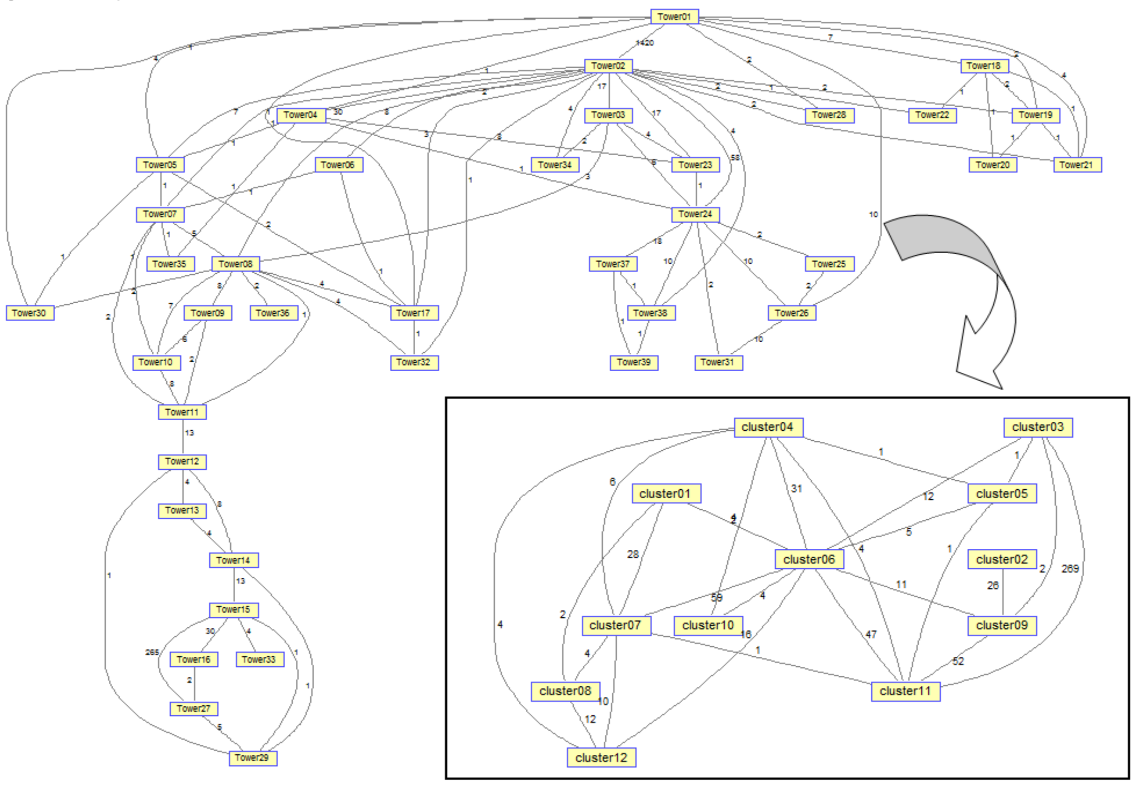

Figure 4: The effect of clustering on the movement pattern before (larger cluster) and after (smaller cluster) clustering

3 Location & Dwell Time Predictions

3.1 Methodology

Assume that we want to predict next location as well as the next dwell-time using a Markov chain of order k. This means that we have observed a sequence of length $2k$ of the most recent k clusters ($a_n, a_{n-1}, ..., a_{n-k+1}$) and the most recent k dwell-times ($t_n, t_{n-1}, ..., t_{n-k+1}$) and we want to predict the next cluster and the next dwell time. Note that variables a_i's (cluster numbers) are discrete while t_i's are continuous. Hence, we have a mixed vector of observations. We treated this problem in two ways:

1- Using quantization and a notion of distance, we convert the problem to a completely discrete problem.

2- We first computed the location prediction without incorporating the dwell times information, and then computed the estimation of dwell time using Kernel Density Estimation. In fact in this approach, we converted the problem into two separate discrete and continuous problems.

We will describe each of these methods in details.

3.2 Location & Dwell Time Prediction Using Quantized Time

Assume that we have observed a sequence of length $2k$ of the last k towers and the last k dwell times. Put each of these observations as a row of $2 \times k$ matrix named P.

- Quantize the dwell times training data into L quantization levels. Since we do not have an algorithm for choosing L, we will choose it heuristically.
- Find a $2 \times k$ matrix Q in the training data whose distance to P is minimum, and replace P by Q (This means that we assume that our observation matrix is now Q instead of P). We will describe our notion of distance later.
- Find the most probable pair $(a_{n+1}, t_{n+1})^T$ in the training data, under the condition of observing Q as past data. In other words, look for the most probable vector $(a_{n+1}, t_{n+1})^T$ which appears right after Q in the training data. We can also use conditional averaging (i.e. the average of dwell times which appear right after Q in the training data) as the prediction of dwell time.

As mentioned above. To choose the number of quantization levels, we did not use any specific algorithm. We simply plotted the location prediction efficiency (percentage of correct predictions over all predictions) of the algorithm on the training data for different values of L and choose the best value of L based on that plot. As shown in Figure 5, $L = 1000$ is a good choice.
There can be several useful definitions for distance. Here we used a distance described as below:
Assume:

$$P = \begin{bmatrix} a_{n-k+1} \cdots a_{n-1} \; a_n \\ t_{n-k+1} \cdots t_{n-1} \; t_n \end{bmatrix}, Q = \begin{bmatrix} b_{n-k+1} \cdots b_{n-1} \; b_n \\ s_{n-k+1} \cdots s_{n-1} \; s_n \end{bmatrix}$$

Then:

$$d(P,Q) = \sum_{i=n-k+1}^{n} (t_i - s_i)^2 + \lambda \sum_{i=n-k+1}^{n} d(a_i, b_i)^2$$

In which $d(a,b)$ is the length of the shortest path between a and b in the towers proximity graph. The intuitive reason of choosing such a distance is that if tower a and tower b are near in the sense of defined distance then there is a short physical distance between them and replacing a with b is somehow reasonable. In fact since we do not have any information about the physical locations of towers, $d(a,b)$ can be a meaningful approximation of physical distance between towers a and b. We also have plotted the efficiency (location prediction efficiency) of the algorithm with respect to k for $L = 1000$. The efficiency of this algorithm in predicting dwell times is not as good as its efficiency for location prediction. The reason is that the data set is not large enough. The raw data is a sequence of tower transitions and corresponding dwell times for a period of a few months. To get rid of the effect of virtual transition due to rapid oscillations between towers, we have performed preprocessing on the raw data (as mentioned earlier). In the new data set, we only have a few hundred transitions for a person during the whole period of experiment (even still some virtual transition is observed). Since the variance of dwell times is too large, as we can see in Figure 6, even the first order conditional mean is not an approximation of dwell time. In fact there is some kind of trade-off in choosing the number of clusters in graph clustering. If we have more clusters, then we will have more virtual transitions which prevent us from making a good approximation. On the other hand, with low number of clusters we might not have that problem, but since the size of each cluster is larger, our precision of prediction will be less. In our experiments for this algorithm, we have assumed 50 clusters of towers.

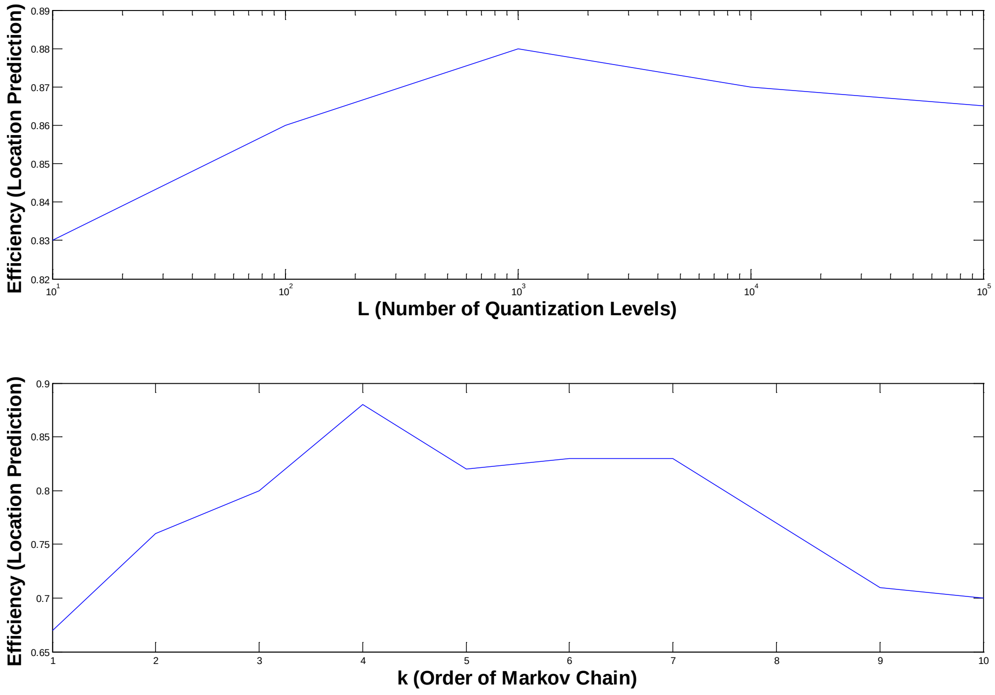

Figure 5: Top plot shows the efficiency of location prediction (ratio of correct location predictions over all predictions) versus L $(k = 4)$. Bottom plot shows the efficiency of location prediction vs. k $(L = 1000)$.

3.3 Location and Dwell Time Prediction Using Kernel Density Estimation

We have made two important observations for dwell time prediction: 1) The current dwell time depends on the current location, as well as the previous locations and the corresponding dwell times. 2) The sequence of dwell times can also be modeled as a Markov chain, and its length should be smaller than length of the location Markov chain.

The formal model of dwell time prediction is as follows:

$\hat{t}_{n+1} = \arg\max_{t_{n+1}} f\left(T_{n+1} = t_{n+1} \mid A_{n+1} = a_{n+1}, A_n = a_n, T_n = t_n, \ldots, A_{n-k+1} = a_{n-k+1}, T_{n-k+1} = t_{n-k+1}\right)$, where k is the Markov chain length for the dwell time prediction and $(a_n, t_n), \ldots, (a_{n-k+1}, t_{n-k+1})$ are the previous k locations and the corresponding dwell times.

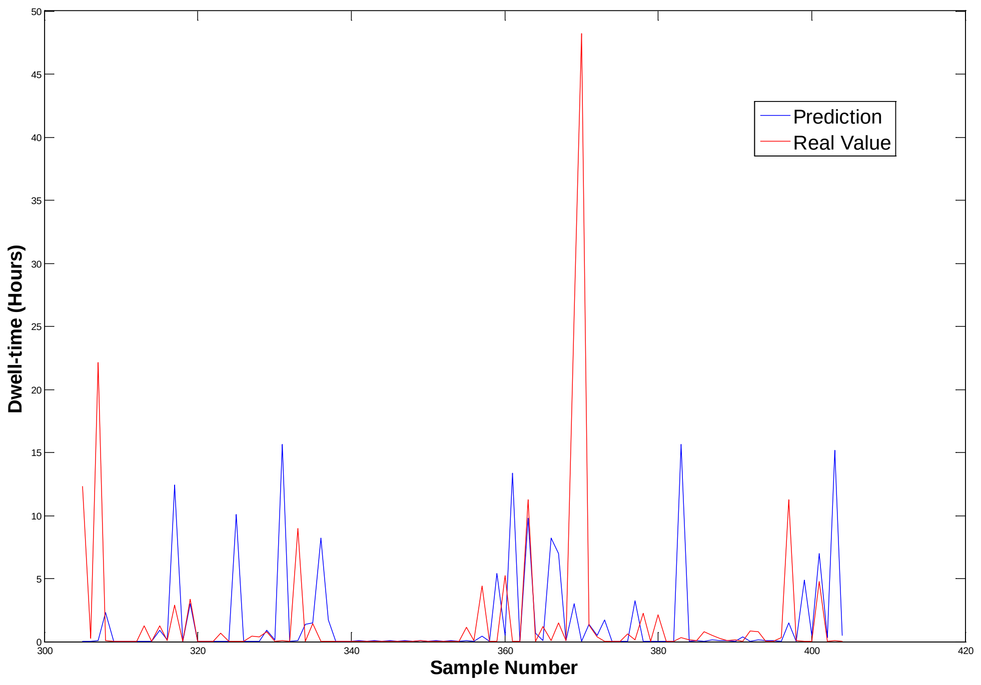

Figure 6: Dwell-time estimation for person number 31 with $k = 4, L = 1000$. *For this experiment we have an average of 2.74 hours error in predicting dwell-times.*

The conditional probability of dwell time can be reduced to a more simple form as follows:

$$f\left(T_{n+1} = t_{n+1} \mid A_{n+1} = a_{n+1}, A_n = a_n, T_n = t_n, \ldots, A_{n-k+1} = a_{n-k+1}, T_{n-k+1} = t_{n-k+1}\right)$$

$$= f\left(t_{n+1} \mid a_{n+1}, a_n, t_n, \ldots, a_{n-k+1}, t_{n-k+1}\right) = \frac{f\left(t_{n+1}, t_n, \ldots, t_{n-k+1}, a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}{f\left(t_n, \ldots, t_{n-k+1}, a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}$$

$$= \frac{f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right) f\left(a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}{f\left(t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right) f\left(a_{n+1}, a_n, \ldots, a_{n-k+1}\right)} = \frac{f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}{f\left(t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}$$

Therefore, the dwell prediction formula can be rewritten as:

$$\hat{t}_{n+1} = \underset{t_{n+1}}{\arg\max} \frac{f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)}{f\left(t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)} = \underset{t_{n+1}}{\arg\max}\, f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)$$

Now the problem boils down to the estimation of the multi-variant conditional probability $f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid a_{n+1}, a_n, \ldots, a_{n-k+1}\right)$ and we decided to use Kernel Density Estimation.

The detailed algorithm for dwell time prediction is as follows:

1) Predict the next location using the Markov chain model:

$$\tilde{a}_{n+1} = \underset{a_{n+1}}{\arg\max}\, \hat{P}\left(A_{n+1} = a_{n+1} \mid H_n\right) = \underset{a_{n+1}}{\arg\max} \frac{N\left(a_{n-k+1} \ldots a_n a_{n+1}\right)}{N\left(a_{n-k+1} \ldots a_n\right)}$$

2) Based on the predicted $\tilde{a}_{n+1}$, estimate the conditional probability $f\left(t_{n+1}, t_n, \ldots, t_{n-k+1} \mid \tilde{a}_{n+1}, a_n, \ldots, a_{n-k+1}\right)$ using KDE. More specifically, the algorithm searches the mobility history H_n for the occurrence of a length-(k+1) sequence $\tilde{a}_{n+1}, a_n, \ldots, a_{n-k+1}$ and records the corresponding dwell times $t_{n+1}, t_n, \ldots, t_{n-k+1}$ as the sample points to feed into the KDE estimator.

3) Since $t_n, \ldots, t_{n-k+1}$ is already fixed, the algorithm sweep the whole sample space of t_{n+1} to find the

optimal solution, the one that maximizes the conditional probability.
We implemented the algorithm as an on-line algorithm which will predict the next time and location and update the old value into history set as next point shows up.
There are a few parameters that are important to the performance of this algorithm. We tuned each parameter on a per person basis and for the ease of explanation we use the data of one specific person as an example to walk through the whole process. After clustering all the cell-phone towers into 50 clusters this person have a sequence of 1739 transitions between clusters (data points). We used the first 1000 data points as the initial history set to predict the next 200 data points.
The first parameter is the Markov chain length. It has an optimal value for each person depending on his/her mobility pattern. If the chain length is too large, it becomes difficult to find the matching pattern that has shown up previously. The '+' line in the Fig. 7 shows the successful location prediction rate as the chain length increases from 1 to 50. It seems that $k = 2$ is the best chain length for this person. Fig. 7 also plots the successful location prediction rate for different initial history size, and it seems that the larger the initial history size, the better the result.
The second parameter is the bandwidth σ for Kernel Density Estimation. Gaussian kernel is used in our experiment. The first figure in Fig. 8 illustrates the effect of σ on the prediction performance. The y-axis is the percentage of points (out of 200) that has a prediction error within 50 percent of its real dwell time. It turned out that σ is not quite important for the performance and a possible reason is that we use $\arg\max_{t_{n+1}} f\left(t_{n+1}, t_n, ..., t_{n-k+1} \mid a_{n+1}, a_n, ..., a_{n-k+1}\right)$ as the prediction instead of mean value.
We applied the algorithm to 15 different people with transition sequence length larger than 1500, and the second figure in Fig. 8 shows the number of points (out of 200) that has a prediction error within 50 percent of its real dwell time for each person. The result for the first person is much better than the others because the parameters were tuned according to this person's mobility history specifically. Considering that most of the dwell time samples are within 0.1 hour the current results has indicated that our idea is worth further research. Developing automated algorithm to tune the aforementioned important parameters according to the observed mobility history can be the next step.

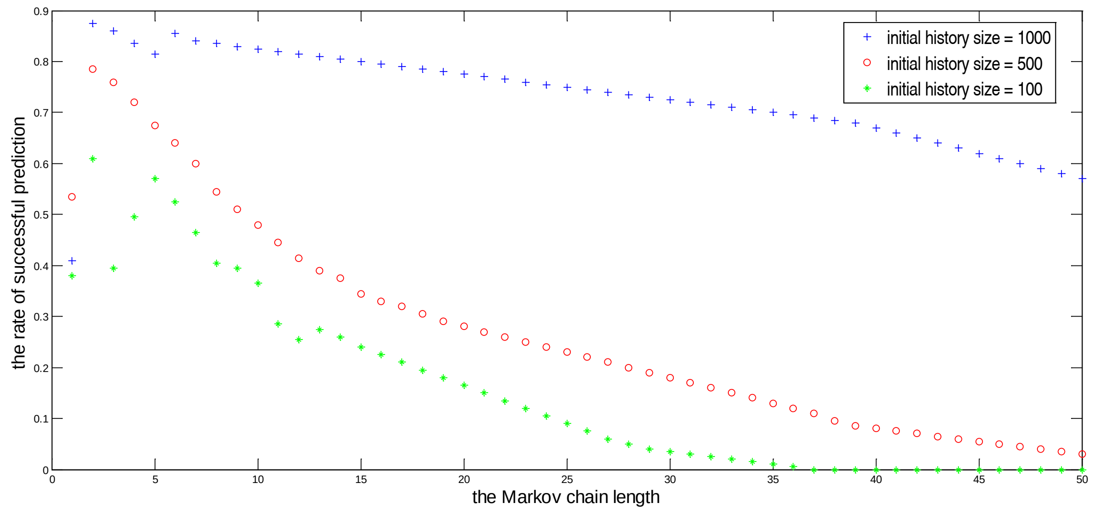

Figure 7: the effect of Markov chain length on location prediction

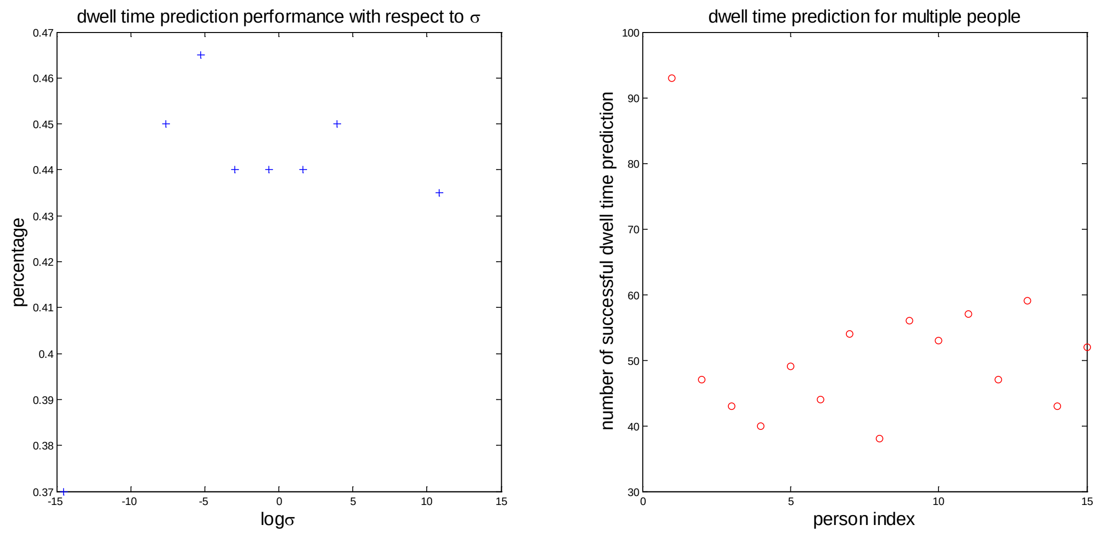

Figure 8: the performance of KDE based dwell time prediction

4 Conclusion

Studying the reality mining dataset, we soon realized that the dynamic and noisy nature of the dataset does not allow the use of raw data directly for predicting subjects' locations. We had to perform some major preprocessing before we were ready to implement any analysis on users' mobility. While portion of virtual transitions due to rapid oscillation between towers was removed in preprocessing, we potentially lost some mobility information as the subject movements were aggregated into clusters of towers. Moreover, the results from the time quantized algorithm were somehow successful in predicting subject's next location. However the performance of that algorithm in dwell time prediction was not as well as its performance in location prediction. The KDE based prediction algorithm using density estimation conditioned on previous time and locations showed reasonable performance on the dwell time prediction. Considering that most dwell times were within 0.1 hour, this result indicated the potential for better performance if we have more balanced dwell time sequence in the future.

Works Cited:
[1] Christine Cheng, Ravi Jain, and Eric van den Berg. "Location prediction algorithms for mobile wireless systems". In M. Illyas and B. Furht, editors, Handbook of Wireless Internet. CRC Press, (2003)

[2] F. Bach and M. Jordan. "*Learning Spectral Clustering*". Neural Information ProcessingSystems 16 (NIPS 2003), 2003.

[3] 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

[4] Samarth H. Shah, Klara Nahrstedt, "*Predictive Location-Based QoS Routing in Mobile Ad Hoc*", Journal of Wireless Personal Communications Volume 21, Number 1 / April, (2002)

[5] 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)

[6] Montgomery C. Douglas "*An Introduction Statistical Quality Control*, John Woley and Sons, Inc. (2005)

[7] N. Eagle, A. Clauset, J. Quinn "*Location Segmentation, Inference and Prediction for Anticipatory Computing*", (2009)

[8] N. Eagle, A. Clauset, J. Quinn *"Methodologies for Continuous Cellular Tower Data Analysis"*, Pervasive (2009)

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