Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Design and Development of Software Agents for Location Privacy-risk estimation

Shashank Sharma

Course of Study: Information Technology

Examiner: Prof. Dr. Kurt Rothermel

Supervisor: M. Sc. Zohaib Riaz

Commenced: June 5, 2018

Completed: December 5, 2018

Abstract

The usage of mobile devices has become ubiquitous in today's world. More people are adopting smartphones to be able to connect with family and friends, easily perform day-to-day tasks, discover new facets, etc. The smartphones are location-enabled and because of this, the rise of Location-Based Applications (LBAs) in the last decade has been phenomenal. Different LBAs use location information for advertising, recommendations, finding new friends, suggesting a new point of interests, etc., based on user trends. However, sharing location data can reveal personality traits, illnesses, political inclinations and religious views. When the location data is collected for a certain period, whereabouts can be easily guessed. For instance, once the location information is collected for a few days, the arrival and departure time from "work" location can be easily estimated. Thus, the characteristics revealed just by location details are narrowly perceived by most users and imposes a privacy risk.

To address this privacy risk, a software agent is developed which helps to estimate and visualize this threat. It can push notifications and aware the user about the privacy risk before he/she decides to share the location with LBAs. In this thesis, we present an algorithm that predicts the user's future movements with confidence percentages. This algorithm processes the raw location coordinates and extracts the meaningful locations. Based on the meaningful locations, a prediction model is formed. This model is used to predict future visits based on the user's current location. The algorithm is evaluated using real-life data from Microsoft Geolife dataset [17] [15] [16]. This dataset contains 182 users GPS trajectory data for a period of 5 years. To inform the user about the privacy threat, a visualization of future visits with confidence percentages is implemented. Finally, a prototype on an Android device is developed to help user estimate the privacy risk before sharing location information on LBAs.

Acknowledgement

I would like to thank my thesis supervisor, M. Sc. Zohaib Riaz, for his valuable assistance, useful critiques, time and suggestions during the thesis. His willingness to give me constructive advice has been very much appreciated. I'm also grateful to Prof. Dr. Kurt Rothermel for allowing me to do my thesis in the department of Distributed Systems.

I would like to extend my thanks to my family members, for their enthusiastic encouragement, and to all my friends for their motivation.

Contents

1 Introduction 13
- 1.1 Motivation . . . 14
- 1.2 Goals . . . 15
- 1.3 Contribution and Thesis Outline . . . 15

2 Related Work 17
- 2.1 Location Privacy . . . 17
- 2.2 Extracting Interesting Locations . . . 18
- 2.3 Next Location Prediction . . . 20
- 2.4 Algorithms Comparison . . . 22

3 System Model and Problem Statement 25
- 3.1 System Model Overview . . . 25
- 3.2 Problem Statement . . . 26

4 Privacy-risk Estimation 27
- 4.1 Location Prediction Model: Overview . . . 27
- 4.2 Markov Chain Model . . . 29
  - 4.2.1 Time-slotted Data . . . 31
  - 4.2.2 Transition Probabilities . . . 33
  - 4.2.3 Algorithm . . . 35
  - 4.2.4 Additional Steps . . . 36
- 4.3 Privacy-risk Estimation Algorithm . . . 37
  - 4.3.1 Approach . . . 38
  - 4.3.2 Proposition: Calibration of Privacy-risk Estimation . . . 39

5 State Formation 43
- 5.1 Variables Used . . . 43
- 5.2 Stay-points Detection . . . 44
  - 5.2.1 Sporadic GPS Trajectories . . . 44
  - 5.2.2 Estimating Arrival and Departure times of Stay-points . . . 45
  - 5.2.3 Algorithm . . . 46

5.3 State Formation from Stay-points 49
5.3.1 Drifting Problem 50
5.3.2 Algorithm 50

6 The PRE Android Application 53
6.1 Objective 53
6.2 User Interface Design 53

7 Evaluation 57
7.1 The Dataset and its Analysis 57
7.2 Evaluation and Results 59
7.2.1 Stay-points Evaluation 59
7.2.2 Time-slotted Data 60
7.2.3 Privacy-risk Estimation Evaluation 64
7.2.3.1 Survey for Memory-loss Factor 65
7.2.4 Prediction Performance 68
7.2.4.1 Metric 68
7.2.4.2 Results 70
7.2.5 PRE Application Performance 71

8 Conclusion and Future Work 73
8.1 Conclusion 73
8.2 Discussion 73
8.3 Future Work 74

Bibliography 75

List of Figures

2.1 Time-based clustering [7] 19

3.1 System Model(left) and PRE Application Components(right) 25

4.1 Flow-chart describing the elements of the Prediction Model 28
4.2 GPS coordinates to Markov Chain Model 29
4.3 Markov chain example on a map 30
4.4 Path modeled using Markov Chain 31
4.5 State transitioning 32
4.6 State weights in each time-slot 33
4.7 State weights normalized 33
4.8 Markov chain derived from state 34
4.9 State transition matrix 34
4.10 Markov Chain Model example with two states 35
4.11 Forming time-slotted data from states 36
4.12 Transition matrix for state 1 and state 2 for time-slot 5 (left without and right with addition of the dummy values) 37
4.13 Steps followed to add noise in the transition matrix 37
4.14 Privacy-risk estimation Algorithm using Markov Chain Model 39
4.15 Steps to apply memory-loss factor to the Markov Chain 40
4.16 Example of application of memory-loss factor 41

5.1 List of Variables 43
5.2 Example 1 of sporadic GPS trajectories 45
5.3 Example 2 of sporadic GPS trajectories 46
5.4 Extracting stay-points from GPS Trajectories 47
5.5 Snapping stay-points to states 49
5.6 Drifting problem while snapping stay-points to the states 50
5.7 Markov chain on states 52

6.1 Android application screen 1 55
6.2 Android application screen 2(left) and 3(right) 56

7.1 Geolife dataset user transportation mode 58

7.2 Geolife dataset trajectory average speed 58
7.3 start and end points from each trajectory 59
7.4 Stay-points count for different time and distance thresholds 61
7.5 User 1 raw trajectory data vs. stay-points extracted 62
7.6 User 1 eight days' time-slotted; 1. Top: Raw Trajectory Data, 2. Middle: Stay-points within trajectories only, 3. Bottom: Stay-points with our approach 63
7.7 User 1 time-slotted data for the November 2008 64
7.8 Path prediction for user 1 65
7.9 User 1 November 2011 hourly distributed data 66
7.10 Path prediction user 1 with minimum confidence of 50% 66
7.11 Path prediction result for user 1 without changes in the algorithm 67
7.12 Path prediction result for user 1 with memory-loss factor of 0.22 67
7.13 Memory-loss factor from survey result 68
7.14 Correct(left), Incorrect(middle) or none(right) prediction scenario . . . 69
7.15 Evaluation for Geolife dataset 71
7.16 Number of states vs Time(sec) on Android Application 71

List of Algorithms

4.1 markovModel() : Create the Markov Chain Model with transition probabilities 35

5.1 extractStayPoints() : Stay-point Detection 48
5.2 adjustStartEndStaypoint() : Adjust start-end time of stay-points 48
5.3 formStates() : Form states from stay-points 51

1 Introduction

Personal computers and mobile devices have become part of everyday life. It is ubiquitous to interact with computers on an everyday basis. Some of these computers are capable of location awareness, for instance, a smartphone. The geographic location can be shared from many sources like Global System for Mobile Communication (GSM), Global Positioning System (GPS), Wi-Fi network location and so on. The most predominant use of location data is by the Location-Based Applications (LBAs) like Google Maps and social networking websites like Facebook, Google+ and Instagram.

There are many LBAs in our mobiles which uses location data for personalized recommendations and advertisements. These are location-based services which runs as applications on user's device. The LBAs uses user location data while the application is active and inactive. The information is often stored on the application's back-end-servers. Every LBA has in its terms and conditions mentioned the usage of the location data which are easily ignored or misunderstood. Consider an example of a weather application forecasting the weather for the next few days on your mobile device. The weather application is using the current location every time the user is moving to a new location for fitting weather forecasts. Another example is of a social networking website like Facebook. The user check-ins and shares location uninterruptedly to find friends and recommendations. On one hand, it enhances quality-of-life by means of the obtained services. On the other hand, the data shared with these applications can reveal private mobility patterns to the service providers as well as to the online social contacts, thus leading to privacy concerns.

For instance, the user shares outdoor movements with an application to receive path recommendations or share some events with friends and family on social networking application. The path reveals three important facts, the source location, the destination location, and the path taken. The source location could be "home", "shopping mall" or a "restaurant" location and so could be the destination. The social events shared can reveal a user's favorite restaurant types or a club/gym he/she has been visiting in the past. When this information is collected for several days, it could reveal the user's "home", "work" and other important locations. Also, it reveals, the time spent at these locations, arrival and leaving times and the transitions from one place to another. This could help, for instance, a restaurant application to suggest a new restaurant, keeping in

mind the user's home location, type of food he/she prefers and his/her time preference of visiting a restaurant.

1.1 Motivation

An overwhelming number of personal characteristics can be predicted online based on location data. It can reveal user's private information like the illnesses based on the specialist he/she has been visiting, vacation locations he/she has been to and political inclinations based on the promotion event he/she has attended.

If the location-based data is collected for a few weeks, it is easy to answer questions like, where does user live/work? Where is the user's club/gym located? What restaurant does he/she likes? Where is he/she at weekends? Which hospital has he/she been visiting? The answers to these questions can give an insight about the user's private life and whereabouts. When this data is collected for few days, the user's movement trends, his/her favorite places, his/her lifestyle, etc., can be precisely guessed. With continuous learning, this data can be updated and have the user's "home" location and "work" location updated with time. This is a privacy attack which makes use of the user's past location data.

Several cases have been emerged in the past where the location data is used unlawfully. Listing one of such examples, Michael Cunningham and his family [14] was tracked for 24 hours for a period of two months using a GPS device. This was initiated as an investigation on whether he filled out his time sheets at work accurately. Cunningham was terminated at work based on the GPS data reports which tracked his whereabouts outside the working hours as well, in which he lied about being at work.

Although in the case of smartphone users, the location sharing is consensual, the privacy risk associated with it is often not well understood by the users. The LBAs use the location data in the background when the application is not even in use. According to a Pew survey [1], 91% of Americans agree that they have lost control over their private data and 64% reported that government must regulate advertisers. It was also reported that only 9% users are confident that social media companies will protect their data.

To understand the privacy threats imposed by LBAs in detail, we must also consider the storage of user location histories on the back-end-servers of such applications. When the location data is continuously collected and stored, a prediction model can be created. This model can make predictions based on a user's current location. For instance, if the user is at "home" at 9 am, the prediction model can reveal his next locations for the rest of the day. Hence, it can be easily estimated where the user would be at any hour of

the day. The predictions can be used to draw inferences on user's daily routines and habits.

The outcome of such predictions can be distributed to third parties, service providers and social contacts. This could be an attack on user's private life. This information can also be leaked on third-party applications and can have unauthorized access. It is easy to oversee the privacy threats that location-based services can impose. With the progress of Machine Learning algorithms and Artificial Intelligence, it has become even easier to exploit this data, understand bulk geographic data and infer meanings from several locations.

1.2 Goals

In this regard, the goal of this thesis is to acquaint users about the consequences of location sharing and the predictability of their future visits. A software application called Privacy-Risk-Estimator (PRE) on smartphone is to be implemented as a solution. This application makes location predictions based on location data previously shared by the user. The future location predictions are shown to the users with confidence percentages. This enables the users to visualize the consequences of sharing location data and understand the privacy risk involved, before sharing the location information on LBAs.

For example, a user wants to share a location data with Google+. The user can use PRE application before doing so. The PRE application will predict the future locations based on user's current location and location history. By analyzing the prediction output, the user can visualize the privacy threats and make an informed decision before sharing the location information with Google+ application.

1.3 Contribution and Thesis Outline

The thesis contributions are:

- Design and development of location prediction algorithm.
- Process the raw location coordinates for stay-point detection and state formation.
- Evaluate the algorithm on real-life location data from Microsoft Geolife dataset.
- Implement the markov chain model algorithm and privacy-risk estimation on android device.

This thesis is organized as follows:

Chapter 2 (Related Work) briefs the related work researched in location prediction and presents the basic understanding of the topics.

Chapter 3 (System Model and Problem Statement) describes the building blocks of the system and the problem statement this thesis is addressing.

Chapter 4 (Privacy-risk Estimation) explains the concepts build for future location predictions and privacy-risk estimation. This chapter explains the aspects for building a location prediction model and use this model to portray the privacy threats to the users.

Chapter 5 (State Formation) discusses the details to form states from raw location coordinates. The concepts and implementations details along with the algorithms are also discussed here.

Chapter 6 (The PRE Android Application) contains the user interface of android implementation and details about the Privacy-Risk-Estimator (PRE) application.

Chapter 7 (Evaluation) contains the evaluation of each individual approach and results of the location prediction model. The outcome of the algorithm on the Geolife dataset is also discussed here.

Chapter 8 (Conclusion and Future Work) concludes this thesis by summarizing the work done, results obtained, open topics and future work.

2 Related Work

In this section, we introduce the related work in the field, the references and motivational work and few parts in detail.

2.1 Location Privacy

The modern computing devices are location-aware. This could pose several privacy threats. The authors [6] surveys how the location-aware computers can be a threat to our private information. The attacker can gain access to this data and reveal many private elements of user behavior.

The authors [6] describes that the information can be gained by first-hand communication and second-hand communication, observation, or inference. The first-hand communication takes place when the user provides the information to the attackers first-hand. An example of first-hand communication is WLAN providing the MAC address. Another example is the location-based services like Google Maps. The second-hand communication takes place when the attacker gains the information from third party. In ubiquitous computing, this is often the case where the information shared with one website is often spread with the others to gain knowledge on user preferences. The attacker may also gain information by observing the user environment. An example of observation attack is the cameras installed in public. The last approach is the inference where with the enough data about the user, inferences can be drawn. For instance, if a user is often visiting a Cardiac Care, he/she has some heart related issues. With enough data, the modern algorithms like Machine Learning algorithms can easily draw inferences based on the data collected. The authors also proposes solutions like policies, limiting first-hand communication or reducing amount of information disclosed to third parties. In this thesis, we propose to limit the first-hand communications.

The topic of location privacy has been discussed by other authors as well. The authors [13] discusses a "location obfuscation" technique in order to make it difficult for the attacker to precisely locate the users. The authors [13] claim that it is still easier for the attacker to retain important information from obfuscated data with high precision. They introduces the approach of using Hidden Markov Model to predict the user movements.

In another paper [12], the author discusses the privacy concerns while using social networking platforms like Facebook. They indicate that there must be more research and implementations for privacy protection as the existing approaches are not enough to curb the user privacy on such platforms.

2.2 Extracting Interesting Locations

In this thesis, we recommend a prediction model which informs user about the privacy risk associated with location sharing. The first step for location prediction is to extract meaningful information from location data. This can take place with first-hand communication. An example of this is Google Maps asking the user to tag locations like “home” or “work”. This reveals directly the most important locations for the user. Another approach of finding the significant places from raw GPS trajectories is inferences. The inferences can be drawn based on few weeks of the location data, which in turn, reveal the locations like “home”, “work”, “favorite restaurant”. This can be done using several clustering algorithms. It is important to understand that the fundamental clustering algorithms like k-mean clustering is insufficient to extract meaningful locations. The typical clustering algorithm do not consider factors like travelling GPS coordinates or short duration stays. We discuss some related work in this subsection for extracting the interesting locations.

Researchers [7] investigate how to extract significant places for the users from raw coordinates data. The author suggests that users are more interested in “places” rather than location. By “places” they mean where the user work/live/play and so on.

Since the Wi-Fi shares the MAC address periodically, hence, the location information is received continuously if the Wi-Fi is connected. The researchers used Place Lab to collect user location data from Wi-Fi enabled devices which works best also for indoor locations. This MAC addresses are then converted into latitude and longitude information with an estimate of 50-100m. The estimation works best in urban areas where the density of access points is high. The authors [7] compared the typical clustering algorithms, k-mean and Gaussian mixture model, on the location data received from Place Lab. Two major drawbacks reported are, the input of the number of clusters in advance, and clusters becoming large comprising of unimportant locations. Knowing the count of clusters or significant locations in advance is difficult as it can vary with users. There are several known algorithms to compute the number of ideal clusters on its own, but it simultaneously increases the complexity of the algorithm. Another issue with these clustering was the increased size of clusters. These large clusters contain unimportant locations because of several transitions between the locations.

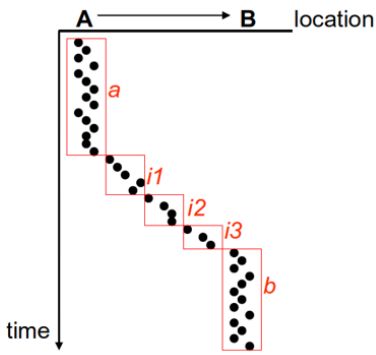

Figure 2.1: Time-based clustering [7]

The authors [7] introduces a time-based clustering algorithm to determine user's interesting locations. The algorithm is depicted in the figure 2.1. The several small dots represent the location coordinates received in an online manner. This algorithm waits for the next location to determine if it belong to the significant place cluster or not. The cluster within a distance threshold which is stayed for at least a given time threshold is considered as a significant place. If the next point is moving geographically away from the cluster mean location, then the new point is not added to this cluster. At this point, the previous cluster total duration stay is checked. If the stay duration is greater than certain time threshold, the cluster is regarded as a significant location, otherwise the cluster is deleted. Hence, this clustering excludes all irrelevant or shortly stayed locations. In the figure 2.1, the clusters a and b are considered as a significant place and the intermediary clusters like i1, i2, i3 are rejected as the duration of the stay was less than the threshold.

The location data is collected from Place Lab and tracked every second. Their [7] results show that the algorithm could extract significant places and works better than k-mean or Gaussian mixture model clustering on location data. The researchers also suggest that the locations must be labelled to extract semantic meaning behind the location coordinates like work/restaurants and so on. The time-based clustering, with some extensions, is also used in this thesis for stay-points extraction.

A research done by [17] also investigate in the direction of mining the interesting locations from location data. The authors aim to extract the interesting locations and classical travel sequences based on GPS trajectory data. Based on multiple GPS trajectories, a Tree-based Hierarchical Graph (TBHG) is created. After this, an approach called Hypertext Induced Topic Search (HIT) is applied.

In this approach [17], a user will be linked to many locations and a location will be linked to many users. The links between users and locations are weighted based on user GPS trajectory data. A hub score is given to a geographical region for a user based on

the travel experiences of the user. It is suggested that a user with a high hub score in a region will visit many places in that region and has a rich travel experience in that region. Considering the hub scores from several users, many interesting places can be mined in the regions. This provides each location within the hub regions an authority score. A user with high travel experience in one region will contribute more to estimate an interesting location in that region.

The researchers [17] system is used by 107 users and the work is also part of the Geolife location dataset from Microsoft [17] [15] [16]. A comparison with rank-by-count and rank-by-frequency is done. Rank-by-count ranks a location interesting if more users have visited that place and rank-by-frequency ranks a location interesting if the location has been visited by the users more frequently. Their approach outperformed these two algorithms. Since this approach requires travel experiences from multiple user location trajectories to extract the significant location for one user, we keep this methodology for potential future work.

2.3 Next Location Prediction

After extracting the interesting locations, the next step is to use them for location predictions. The next location prediction can inspect user's day-to-day locations. This can also suggest user's future placements. Research was done by [11] in the field of next place location prediction which suggests the next predicted location based on user behavior. The main idea is to use user check-ins on Foursquare to predict user movements. The data of 35 million check-ins from across the globe over the period of 5 years is used. The idea is explained how user check-ins not only allows us to see the locations user visited in the past but also help us understand the mobility patterns of the users. They have used prediction features like user preferences, the popularity of the places and geographic distance between places. On these features, they have applied supervised learning with linear model and M5 model trees.

One of the first tasks addressed [11] is the next check-in prediction. The next check-in is predicted based on the current check-in data and several other factors. First, all the possible next location check-in based on current check-in are ranked. It is suggested that 99% of the next check-ins are within 10 kilometers radius from the current check-in. The check-ins are also mostly in urban areas. The ranking is performed based on historical visits by the user to a place, categorical preferences based on what category of places have user checked-in in the past and social filtering based on where user's friends are checked-in. The next task is a global mobility feature to determine check-in patterns irrespective of user preferences. This uses the popularity of the geographic location, geographic and relative distance of all the other locations from user's current

location and activity transition where few locations are visited after specific locations, for instance going to a hotel after an airport or railway station visit. Next, they assign the temporal features to each place. Based on the hour category, what type of place has been checked-in in a particular hour of the day or week?

After the assignment of these features, the ranks(k), percentile rank (PR) and average percentile rank (APR) are defined for each venue. The highest rank is given to the location where the next check-in could take place. The analyses from the researchers suggest that APR is scored higher for categorical preferences with 0.84 when compared to historical visits with APR 0.68 and social filtering with APR 0.61. In global mobility section, place popularity has better APR which is 0.86. Activity transition features also achieve only 0.60. The study [11] also suggested that people tend to stick to their set of location check-ins during the day time but visited new locations during evening. All these features suggest that there could be many factors which can affect user movement patterns. They finally used all these features and combined them into a supervised learning framework. With M5 tree, they have received an APR of 0.94 and linear regression model only resulted with an APR of 0.81 which is less than many individual feature APR. The authors explained how the prediction model can have better performances with several features combined. The evaluation is done on a social networking check-in dataset and hence, this could have influenced the evaluation results.

Authors [5] also discusses mobile based next location prediction based on current location. They suggest using contextual data along with spatial and temporal data associated with location. The mobile call/SMS logs, accelerometer and Bluetooth can have additional information which has not been investigated before for location predictions. The researchers explain how location prediction is very user specific, the data is evolving with changing city/work location, etc. and it is possible to have missing location data.

The model [5] pre-process the raw data which keeps the short-term data with its contextual information. The model should accept and integrate new location data and an updated check of actual next location vs. predicted next location is performed to keep an updated accuracy rate. This model is implemented in an online manner on a mobile device. The data used as an input is from Nokia Mobile Data Challenge (MDC) [9] from 200 participants over one year. First, the raw data is processed to extract temporal features, phone status, phone usage and other features. These features include the hour and time duration of a visit, the ring-tone used, last call/SMS log and others. Then a classification technique is used with a software named WEKA.

The results [5] suggested that the regular users were easy to predict with an accuracy percentage of 80% whereas the users with irregular movement patterns are difficult to predict. In feature selection phase, it has been found that keeping all the features gives

the best results with accuracy of 92%. The contextual information addition is added as a potential future work of this thesis.

The paper [5] also suggests an alternative advertisement approach. For instance, a user will be more interested in dinner promotion/discount before he/she goes outside on dinner time. The user can share the location-based data with the telecom provider, and the telecom service provider can act as an intermediary between the user and the restaurant advertisement company. The third party like, in this case, the restaurant company, can push the relevant advertisements on the user's phone based on the predictions result shared by the telecom provider, without disclosing user's information or location data to the third parties. In this scenario, the user can receive more relevant advertisements and still have not shared his private information with advertisement companies. This can help in preventing personal data to be shared with companies and third parties and hence, preserving user privacy.

The authors [2] design hierarchical hidden semi-markov-model concerning spatio-temporal location data to predict human mobility patterns. In this model, each state denotes either a stay-point or a transition from one place to another. The states which are more visited will be super states consisting of other states, and the states geographically closer or spatio-temporally closer are more likeable to be in one state. Hence, there are super states which contain other states. The next step is to map each location coordinate in a grid with cell id. The algorithm becomes expensive with increasing states. The states can be reduced by using states at a higher level in the hierarchy. This, in turn, reduces the total complexity of the algorithm. The researchers have used real-life location data Geolife dataset [17] [15] [16] and Capricorn dataset [3] [8]. The approach has better results in the presence of noise and missing data.

2.4 Algorithms Comparison

The author [4] compared 18 different location prediction algorithms. The focus here is on the accuracy of prediction along with other parameters to compare these algorithms with each other. Based on their analysis and knowledge gained during the algorithm comparison, they also present a new next-place prediction algorithm called MAJOR. The dataset used by the researchers is Nokia Mobile Data Challenge (MDC) [9]. It contains 37 user mobile phone data over 1.5 years.

They [4] have considered several spatial and temporal features with different combinations and named each algorithm based on the features used. For instance, the features used are, current location of the user P1, current and previous location of the user P2, time of the day H, Day of the week D, weekday or weekend W. Here P1 and P2 are

spatial features and H, D and W are temporal features. Using the combination of these spatial and temporal features, they have formed several algorithms e.g., DP1, WHP2 and so on. On these algorithms, several performance metrics are calculated, which contains factors like accuracy percentage A1 which is the ratio of correct predictions to the total predictions. Other performance factors which included the true positive, false positive, true negative and false negative with respect to transitions. For example, a true positive transition is the transition which is correctly predicted from one place to another. True positive transition rate TTPR is the ratio of true positive transitions over the total transitions. Similarly, other rates like false positive transition rate, are calculated. Some other interesting performance parameters included transition precision ratio which is calculated as ratio of number of correctly predicted transitions and the total predicted transitions. The researchers also considered the arrival and departure events prediction from a particular place as a performance metric.

Using the combination of different spatial and temporal features, they [4] have compared 18 different prediction algorithms for their predefined performance metrics. The first comparison is highlighted for algorithms considering only spatial features, only temporal features or both together. Most algorithms which can achieve a good prediction accuracy fail to predict a transition and vice-versa. This concludes that there exists a trade-off between prediction accuracy and transition prediction. This is overcome by the novel approach introduced in the paper called MAJOR. This new approach runs all 18 algorithms (spatial and temporal combinations) together and selects the one with the highest votes. This means that, if the highest vote among the 18 algorithms suggests a transition, then the transition is predicted, otherwise not. The same voting approach is applied for the prediction of next place. This gave MAJOR an accuracy of 82% but only 21% detection of true transition. To improve the transition detection ability, they have introduced a voting threshold. The analyses suggested that a median of 8 approaches predict a true transition and a median of 3 approaches predict a transition when no transition occurs. This will help to decide the voting threshold offline. If the minimum number of approaches voting for the transition is greater than or equal to the threshold, then it is considered a transition, otherwise not. The researchers [4] help us to understand the different metric which is important in location prediction algorithms.

3 System Model and Problem Statement

This chapter consist of the system model describing the system components and present the problem statement addressed in this thesis.

3.1 System Model Overview

Our system model comprises two components, namely: mobile phone and location-based services as shown in figure 3.1. An assumption is made that the user makes the location sharing decision manually.

The mobile phone is equipped with location determination technologies like GPS. This smartphone is capable of outdoor location tracking. The mobile phone exchanges information with several location-based services.

This mobile phone also runs the Privacy-Risk-Estimator (PRE) application. The location data is updated and shared regularly with the PRE application. PRE informs the user about his/her predictability and privacy threats related to location sharing. Hence, the user can make an informed decision before sharing location information with other location-based services. The decision is made by the smartphone users and hence, giving the users more authority on their privacy. The decision can protect the user privacy and attack on his/her location data.

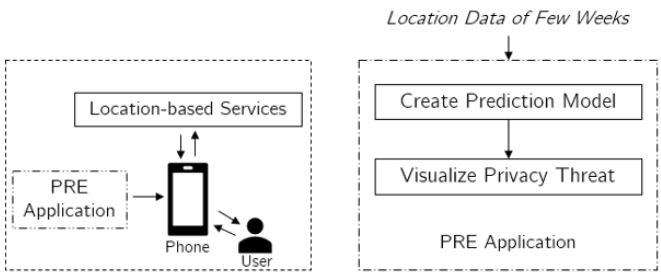

Figure 3.1: System Model(left) and PRE Application Components(right)

The components of the PRE application are shown in 3.1 on the right. The application receives location data for a few weeks. This data is used to create a prediction model. This prediction model keeps a track of user visits and transitions from different locations. Finally, the prediction model is used to visualize the privacy threat. This visualization includes the future locations predictions based on current location. In other words, it can aware the users about the consequences of sharing the location with other LBAs.

For example, a user named Carl wants to share his current location gym with Google+ application. He also has PRE installed in his smartphone. Before sharing the location with the Google+ application, Carl checks the privacy risk involved using PRE application. PRE application suggests Carl all the locations which are predictable after gym, based on his location history. The estimations suggest that Carl would visit a particular restaurant in the coming hours and then he would go back to his home location. Carl can now decide if he want to continue sharing the gym location with Google+ or not.

3.2 Problem Statement

The threat of location prediction and exploitation of user privacy is to be shown to the users so that they can make a wise decision before sharing the location with third-party applications on mobile devices. In this regard, we define the following requirements:

1. An algorithm should be developed which can make realistic and accurate future location predictions based on location history of the user.
2. These predictions should be available for privacy-risk estimation locally on user's smartphone.
3. The privacy threat should be visualized in a way that it is easily understandable by most users irrespective of their educational background.
4. The detail of the privacy risk should be realistic. User should be able to calibrate the risk estimates.
5. The algorithm should be simple enough to be able to run on a mobile device with limited CPU and memory.

4 Privacy-risk Estimation

In this chapter, a privacy-risk estimation approach is discussed. A location prediction model (markov chain model) that predicts the future visits of a user based on their current location is examined. This model is then used for privacy-risk estimation. The idea is to raise awareness to users about the privacy risk while sharing the location with applications on mobile phones.

4.1 Location Prediction Model: Overview

The human mobility pattern can be dependent on several features like user's occupation. For instance, if the user is a salesperson or has an occupation which requires daily travel, it is very unlikely that the user has a regular "home-work-home" pattern. These users are difficult to be predicted. Oppositely, users who have very regular movement patterns are easily predictable. The idea is to have a prediction model which can work for everyone.

The user tends to have a pattern where the next location is dependent on their current location. For example, the "work" location is often visited directly after "home" or in many cases, after the visit to the "supermarket", the user tends to go back to "home." These trends could be often predictable but sometimes they are not obvious. For instance, a "restaurant" visit could occur after "home" or "work" visit or even after a "shopping mall" visit. But most people have a pattern of transitioning between the places, where the next visit is dependent on their current location. To keep the model simple, a markov chain prediction model is suggested. This model makes predictions based on current location only.

The prediction model contains several steps. The flow chart depicted in 4.1 explains how the model works. The location coordinate data with the date and time information is fed as input to the model. The model receives the location coordinates in an online manner. This is to simulate how user shares the location details with other LBAs like Google+, Facebook, etc. The model keeps collecting the GPS points till the end of a time-slot, which is depicted using the "hour change" decision block. The prediction model works on discrete time-slots t. These time-slots are divided based on the hours of

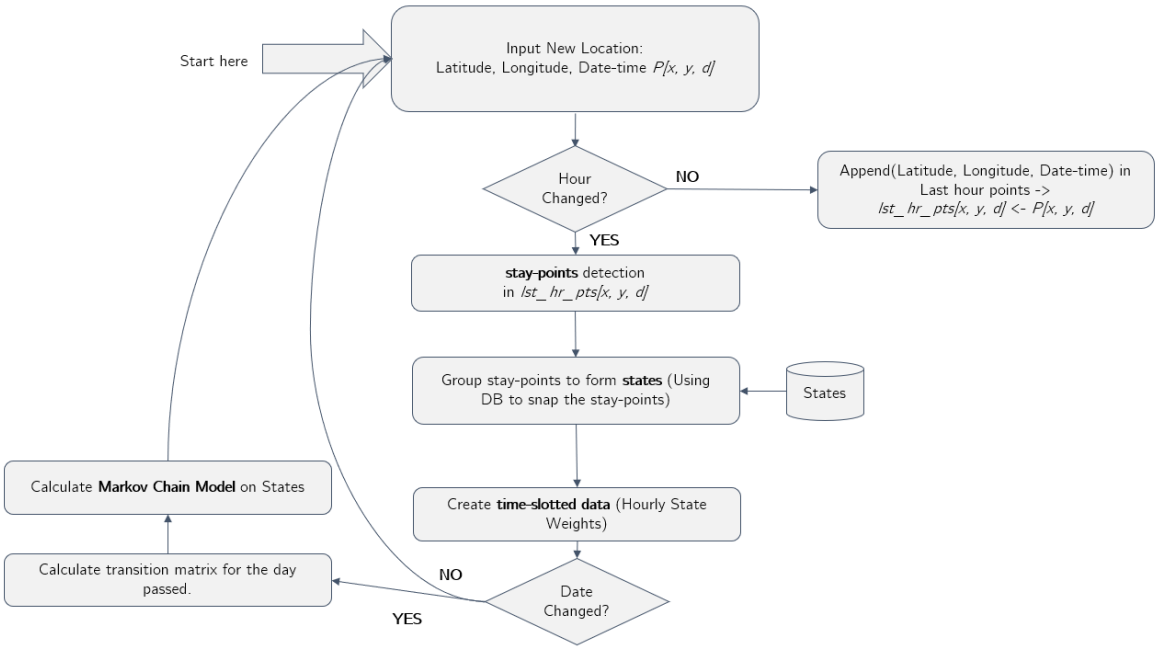

Figure 4.1: Flow-chart describing the elements of the Prediction Model

the day. Hence, time-slots t ranges from 0 to 23. After the end of each time-slot, several steps are performed on the points from the previous hour. These steps are:

- Stay-points detection (discussed in chapter 5)
- Group the stay-points to form states (discussed in chapter 5)
- Create time-slotted data (discussed in current chapter 4)
- Apply markov chain model (discussed in current chapter 4)

The location coordinates P[x, y, d], where x, y and d are latitude, longitude and date-time information respectively, are received and after each time-slot, significant locations called stay-points as $sp = \{sp_1, sp_2, \ldots\ sp_n\}$ are extracted. These stay-points are the locations where user has spent significant amount of time. This step removes the noise (travelling locations coordinations or short stay locations like post-office visit) and makes sure that the model is built on stable and longer stayed locations only. As a next step, the stay-points are clustered together to form states as $st = \{st_1, st_2, \ldots\ st_n\}$. The states are formed by combining the similar stay-points based on their geographical distance from each other. These states are used to form the time-slotted data and finally creating the markov chain model.

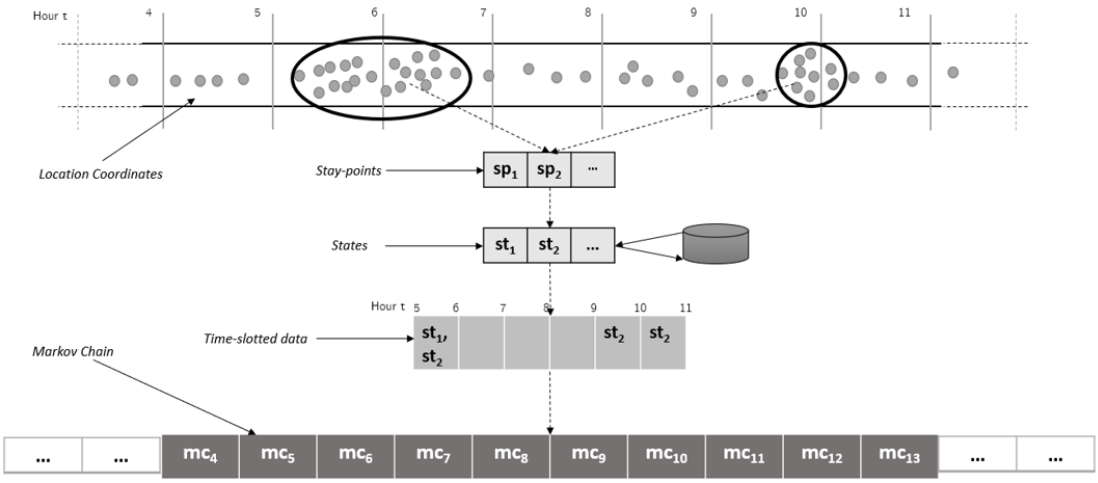

Figure 4.2: GPS coordinates to Markov Chain Model

A more detailed depiction is done in figure 4.2. The top layer depicts the incoming location points. The longer stayed locations are tracked down as stay-points as shown in the second layer of the figure 4.2. The stay-points are combined, and the states are formed as the next step. These states are then weighted within their corresponding time-slot which forms the time-slotted data.

Please note, this is just a quick introduction of the different elements of the prediction model. In this chapter, we introduce the prediction model and privacy-risk estimation algorithms in detail, assuming that the states are already available. The formation of states from raw location coordinates is described in further chapters.

4.2 Markov Chain Model

The markov chain model for location prediction is formed based on states. In this chapter, we assume the states are already available.

The model of markov chain is created to calculate the probability of going from one location to another. The figure 4.3 shows on a geographical map how the markov chain looks like. There are four important locations marked on the map. These marked locations are called states. The markov chain model will contain the transition probabilities from one of these important places to all the other places (including self) as depicted in the figure 4.3. The arrow depicts the transitions from one location to another (including self). It is important to understand that the model proposed in this thesis considers only the significant locations and the transitions between them. Hence, the travelling

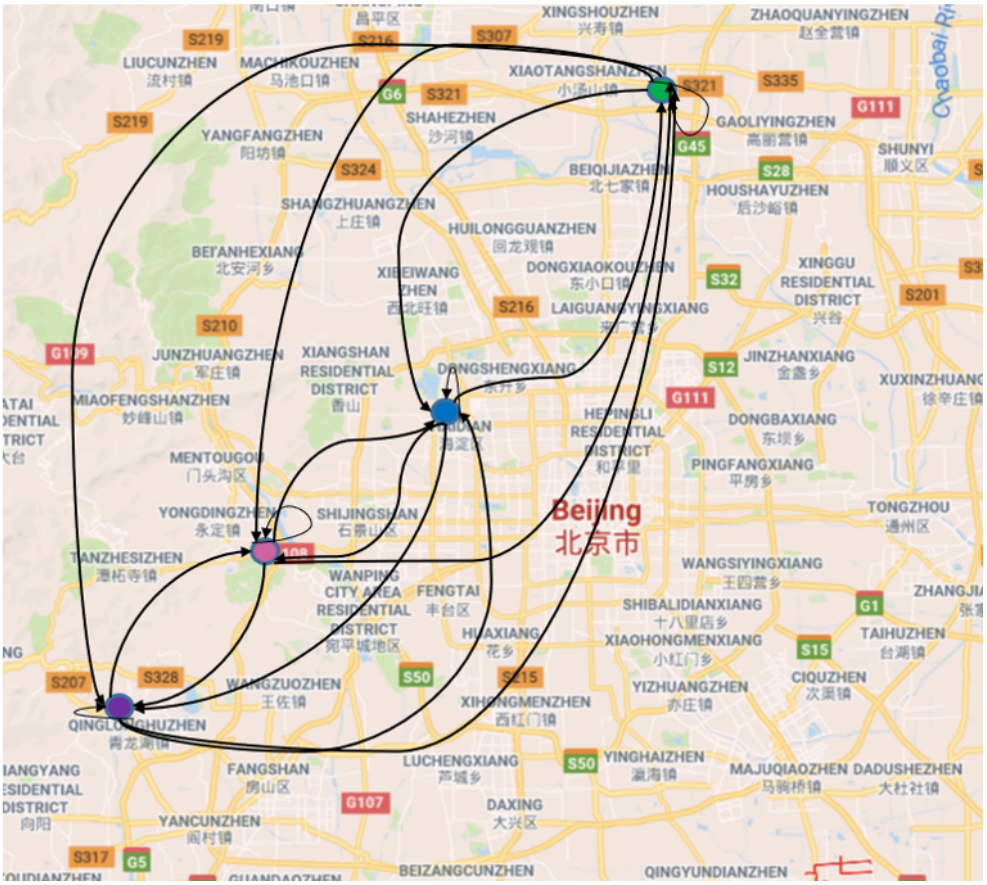

Figure 4.3: Markov chain example on a map

location coordinates will be ignored as noise and will not be part of the markov chain model. The extraction of these important locations from the raw location coordinate points is detailed in the further chapters.

Once the model is created, the transition probabilities for each time-slot are available. Based on the transition probabilities for each time-slot, a path can be predicted. A path is the series of locations visit in the coming time-slots. The path prediction uses the current location, current time-slot and the markov chain model. The example is extended on a geographical map as shown in the figure 4.4. For instance, the model can suggest, if the user is found to be at state 1 at 7am, the next visits will be at location 2, 3, 4, 3, 1 at corresponding time-slots as shown in the figure 4.4. This results into a path which is predicted using the markov chain model. The green arrow depicts the transition from state 1 to 4 and the red arrows depicts the transition from state 4 back to 1. Each next visit is predicted with a corresponding confidence percentage which is derived from the markov chain model. The path prediction is done to indicate the predictability of a

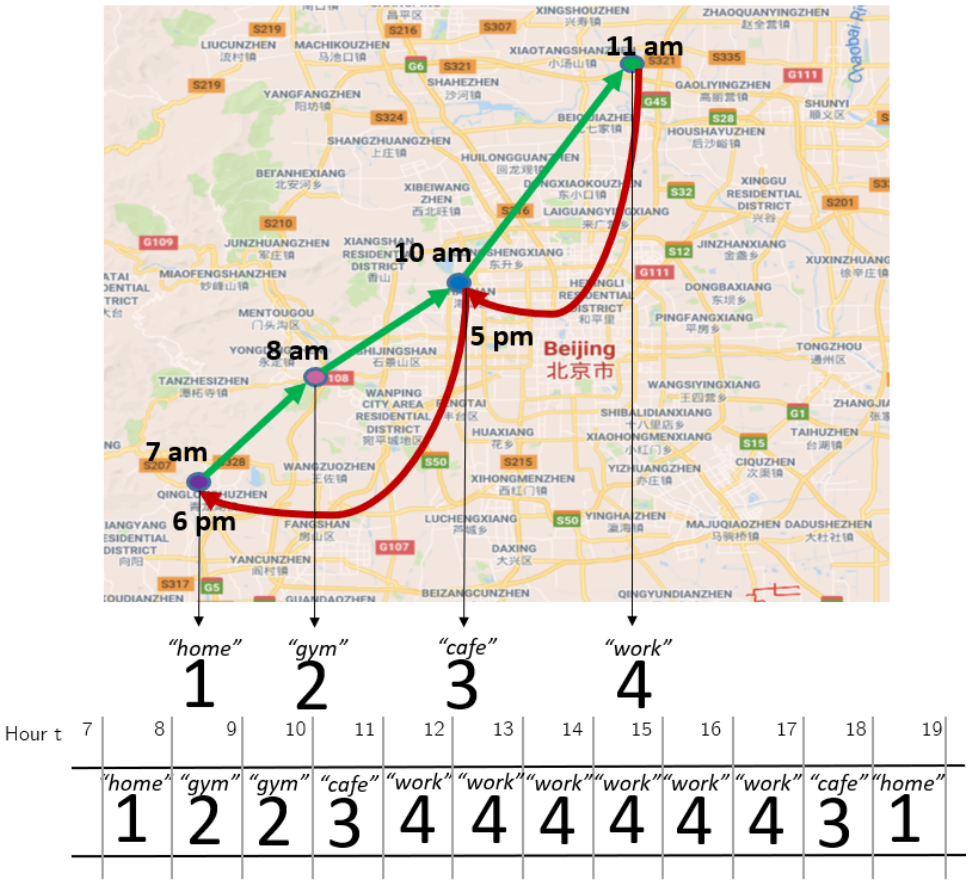

Figure 4.4: Path modeled using Markov Chain

user knowing the current location, current time-slot and past location trends. The path prediction is detailed in further sub-chapters.

4.2.1 Time-slotted Data

Since, we assume that the states are already available at this moment, we continue the discussion of state processing. The next step is to create time-slotted data. The time-slotted data is nothing but the state weights in discrete time-slots.

The states $st = \{st_1, st_2, \ldots st_n\}$ are used for calculating the weight $w = \{w_1, w_2, \ldots w_k\}$ of each state in each time-slot. It is important to know that there can be many states in one time-slot. The weight w_i represents the stay at state st_i in a time-slot. Since states itself carry some semantic meaning like "home", "work" or others, the weights w represents how long did the user stay at "home" in this time-slot.

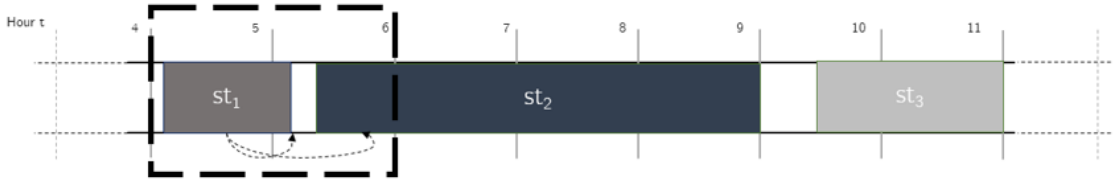

Figure 4.5: State transitioning

The same state can appear in different time-slots. The set of states st = $\{st_1, st_2, \ldots st_n\}$ are assigned with weights at each time-slot to form $w^t = \{w_1, w_2, \ldots w_n\}$ for time-slot t, where w_1 to w_n represents the weights of states st_1 and st_n respectively. In our case, time-slots are divided into hours of the day. Hence, the hourly weights of each state form the time-slotted data. Hence, we use the time-slotted data and hourly state weights interchangeably.

To understand this in detail, consider an example as shown in the figure 4.5. There are three states st_1, st_2 and st_3 which exists between hour 4-6, 5-9 and 9-11 respectively. We consider the time slot between hour 4-5 as 4 for simplicity. Hence, we have a total of 24 time-slots ranging from 0 to 23 in a day, representing each hour of the day. The vertical drawn lines represent the hour of the day and the width of a state rectangle represents the duration of the stay at that state. For instance, st_1 stay is recorded at time-slot 4 and 5. The states can exist in more than one time-slot, like st_1. There can be many states in one time-slot, like st_1 and st_2 in time-slot 5.

Let us consider the example of transitioning from hour 4 to hour 5 as depicted in figure 4.6. In this example, the state transition from hour 4 to hour 5 is from st_1 to st_1 and st_1 to st_2, represented by dotted arrows. The state weights represent the duration of stay at the state in that time-slot. The weight of state st_1 is 60/60 as the duration of stay at st_1 in time-slot 4 is for 60 minutes. Similarly, the weights of st_1 and st_2 are calculated in time-slot 5, which is also indicated in the figure 4.6.

It is important to mention that the hourly weights are normalized before the markov chain is calculated. The normalization is done by dividing the weights of all the states in a time-slot with the sum of all the weights in that time-slot. Let us consider the time-slot 5 in our example figure 4.6. The state weights for state st_1 = 0.17 and for state st_2 = 0.58 in time-slot 5 are divided with the sum both the weights i.e. 0.75. After the normalization step, the states are shown in the figure 4.7. This is done to smooth the data in each time-slot. The sum of the weights in time-slot 5 is now 1.

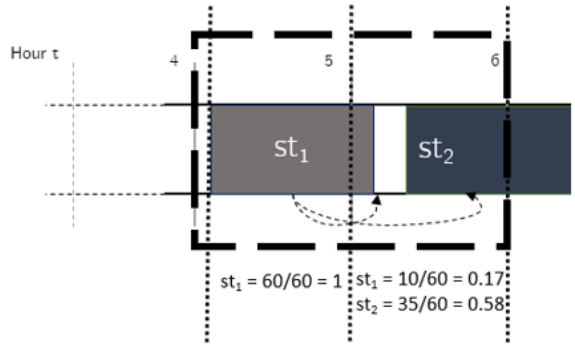

Figure 4.6: State weights in each time-slot

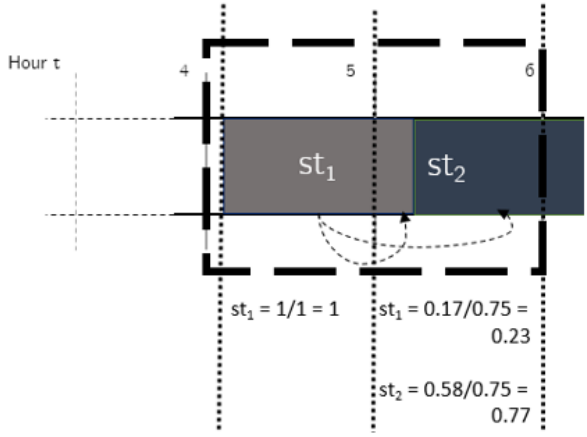

Figure 4.7: State weights normalized

4.2.2 Transition Probabilities

Creating the markov chain model involves calculation of the transition probabilities. The transition probabilities indicate the chances of transitioning from one state to another. The markov chain holds the probability of transitioning from one state to another, in discrete time-slots. The time-slots are represented by hour of the day, as shown in figure 4.8. Hence, the markov chain records all the transitions from hour t to hour t+1.

The markov chain built is standard and the probability only depends on the current location and not the locations before. The model is updated after each day to keep a track of all the new locations user has visited in the previous day and update the mobility pattern. The continuous update of the markov chain model helps to track the changes in user behavior. For instance, user may change the work location, working hours, join a new gym or move to a new city.

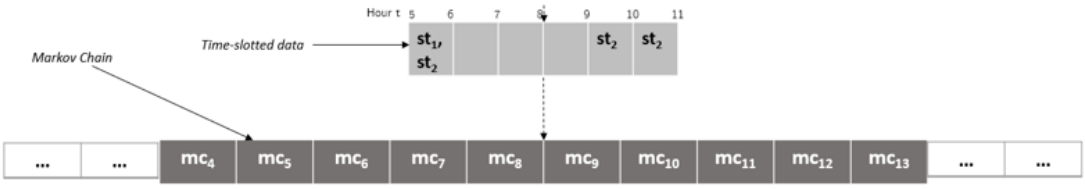

Figure 4.8: Markov chain derived from state

Figure 4.9: State transition matrix

The states st = {st_1, st_2, ... st_n} has corresponding probabilities p = {p_1, p_2, ... p_n} in a time-slot. Let us continue the previous example from figure 4.7. The probability of transitioning from st_1 to st_1 from hour 4 to 5 is calculated based on the hourly weight of st_1 in hour 4 and hourly weight of st_1 in hour 5. Similarly, the probability of transitioning from st_1 to st_2 from hour 4 to 5 is calculated based on the hourly weight of st_1 in hour 4 and hourly weight of st_2 in hour 5. The weight vector w^t = {w_1, w_2, ... w_k} represents the weight of states from state 1 to k for time-slot t. The weights vector w^4 = {w_1, w_2} for hour 4 can be defined as w^4 = {1, 0}, where the 1 represents the weight of state st_1 and 0 represents the weight of state st_2 in time-slot 4. A similar weight vector w for the next time-slot 5 can be represented as w^5 = {0.23, 0.77}. The multiplication w^4(transpose)* w^5 results into a matrix as represented in the figure 4.9. Several such matrices are generated for each time-slot. For each time-slot, there are also many matrices generated on different days. The matrices for same time-slots are added. The resultant matrix rows are normalized which results into the sum of each row as 1. The matrix now represents the transitions probabilities among states st_1 and st_2 from time-slot 4 to time-slot 5. The first row represents the transition probabilities from state st_1 all the other states i.e. st_1 and st_2. Similarly, the second row represents the transition probabilities from state st_2 to all the other states. This represents now a markov chain model for the given time-slot which includes all the known states. The markov chain for transition from hour 4 to hour 5 is called mc_5 which is depicted in the figure 4.10. The arrows indicate the transition probabilities from one state to another. A transition can be a self-transition, for instance, st_1 to st_1 or a transition to another state, for instance, st_1 to st_2. These transitions represent human mobility from one important place to another. To understand this process in detail, please refer the paper [13].

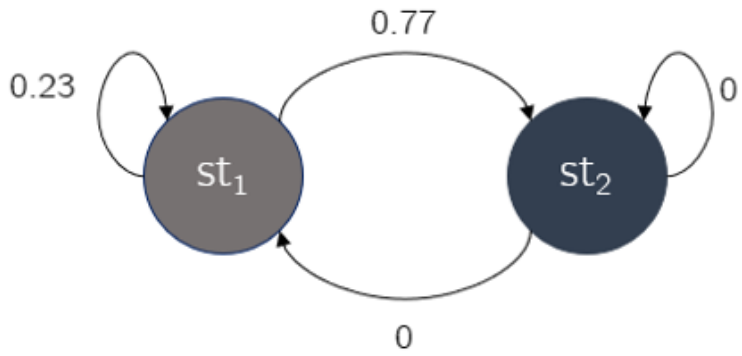

Figure 4.10: Markov Chain Model example with two states

4.2.3 Algorithm

Algorithm 4.1 describes the creation of markov chain model. After state weights w = $\{w_1, w_2, \ldots\ w_k\}$ are calculated, the next step is to build the markov chain model mc. For each t hour of the day, the transposed weight vector w^t is multiplied with the weight vector w^{t+1}. This gives the transition matrix trn_mat_{t+1}. The transition matrix for time-slot t is added with the transition matrix of the same time-slot t, for all the days. To keep it understandable, this step is not shown in the algorithm. The resultant transition matrix trn_mat_{t+1} is used to calculate the markov chain by dividing each cell element of the matrix with the sum of the row. This converts the cell elements into probabilities. The sum of each row is now 1. Note, for the last time-slot of the day, i.e. 23, the weight for time-slot 23 and the weight of time-slot 0 for the next day is used. To keep it simple, this next day is also not indicated in the algorithm.

Algorithm 4.1 markovModel() : Create the Markov Chain Model with transition probabilities

Input: State Weights *w*
Output: Markov Chain Model *mc*

1: **for** each $t - hour$ from $0 to 23$ **do**
2: **if** $t! = 23$ **then**
3: $trn_mat_{t+1} \leftarrow w^t[T] * w^{t+1}$
4: $mc_{t+1} \leftarrow eachCell(trn_mat_{t+1})/rowSum(trn_mat_{t+1})$
5: **else**
6: $trn_mat_0 \leftarrow w^{23}[T] * w^0$
7: $mc_0 \leftarrow eachCell(trn_mat_0)/rowSum(trn_mat_0)$
8: **end if**
9: **end for**

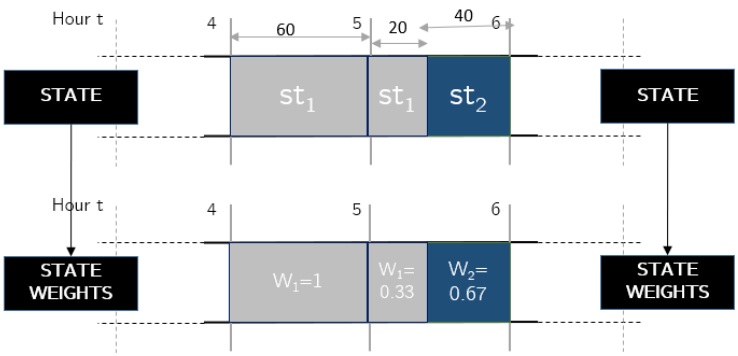

Figure 4.11: Forming time-slotted data from states

4.2.4 Additional Steps

Once the markov chain is calculated for each time-slot, some additional steps are performed. Consider the figure 4.11 as an example. The weights of state st_1 and st_2 for hour 4 and 5 are depicted in this figure. There is a transition from hour 4 to 5 from state st_1 to st_1 and another transition from st_1 to st_2. The corresponding state weights represents the normalized duration of stay in that time-slot. Hence, the w_1 is 1 at time-slot 4, because the entire stay at this time-slot is at state st_1, w_1 is 0.33 (20/60) between time-slot 5 and 6 as the stay at state st_1 between 5 and 6 hours is 20 minutes. Similarly, the weights of the other states are calculated.

The transition matrix for time-slot 5 is shown on the left part of the figure 4.12. The rows and the columns are representing each state and each cell represent the transition from one state to another. The first row represents the transition from st_1 to all the other states i.e. st_1 and st_2. To calculate the probability of transitioning from st_1 to all the other states can simply be calculated by dividing each cell elements with the sum of the row. If the sum of the row is zero, this indicates that we have no information about this transition. In this case, equal probabilities are assigned to all the states. This is shown on the right of the figure 4.12, where the transitions from state st_2 are assigned with equal probabilities.

There could also be cases where only few cells in a particular row of the transition matrix are 0. In this case, the sum of these rows will still result in 1 and not in 0. In such cases, the 0 cell values are replaced with a very small noise value. This is done to ensure that the chances of transitioning from a known state to another known state is never 0. The figure 4.13 explains the process of replacing the zero probabilities. The first step shows the transition matrix calculated as discussed earlier. This transition matrix has a cell value 0 for transitioning from st_1 to st_3. In step 2, 0 is replaced by a 0.00001, a small

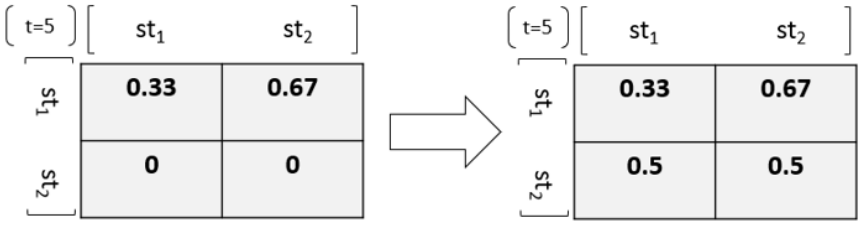

Figure 4.12: Transition matrix for state 1 and state 2 for time-slot 5 (left without and right with addition of the dummy values)

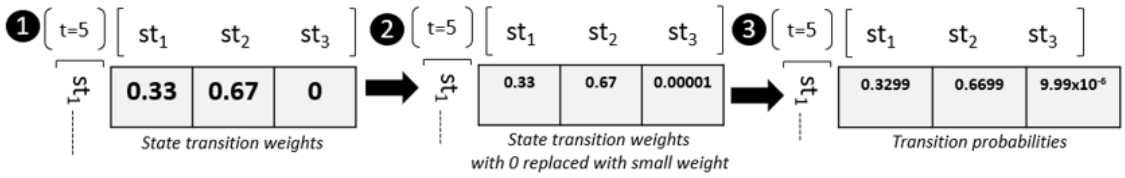

Figure 4.13: Steps followed to add noise in the transition matrix

weight. In step 3, each cell value is divided by the sum of the row, making the transition probability from state st_1 to st_3 non-zero.

4.3 Privacy-risk Estimation Algorithm

The privacy-risk estimation is to inform the user about his/her predictability. The idea is to predict several paths that user may take, from a known location, in future time-slots. For example, user's current location is known to be at "work" at 8 am. Knowing his/her location data for few weeks, the markov chain model is built. The privacy-risk estimation can be done using markov chain model. Now this markov chain model is used to predict all the locations in consecutive time-slots. One of the paths predicted could suggest that he will stay at "work" location till 5pm with very high confidence. Another path could suggest that he will go back to his "home" location at 3 pm with medium to low confidence. Each next location is predicted with a confidence percentage. There could be several such paths that could be predicted.

The paths are predicted only based on the markov chain model. If the model is not recent or is built with very limited data, the predicted paths may not make much sense. For example, the markov chain model is build based on the 2 days data where user transitioned between "home", "shopping malls" and "restaurants" only. The two days do

not include “work” location. It could have been a weekend, public holidays or user has taken a vacation. Now, on third day, if the paths are predicted, only these locations are taken into consideration and the paths could be misleading if user started working from day 3. Hence, the markov chain model must be built on several days location data and the data should be recent to make improved predictions.

4.3.1 Approach

The markov chain model contains the transition probabilities from one state to all the other states. The states st = $\{st_1, st_2, \ldots st_n\}$ has corresponding probabilities p = $\{p_1, p_2, \ldots p_n\}$ in a time-slot. These state probabilities are used for privacy-risk estimation. For this, a state (current location) and a time-slot (current hour) is required as input. Based on these inputs, the future locations are predicted to showcase the privacy-risk to the users.

Each next location is predicted with a confidence percentage. We define a confidence threshold value which can range from 10-100%, since the confidence of 0% would not produce any meaningful results. The prediction stops one step after the confidence drops less than the confidence threshold value.

Consider the figure 4.14, as an example figure for markov chain model. The start state is i with a probability of p_i at time-slot t. The predictions are made for the future time-slots i.e. t+1 and further. At each time-slot, there exists probabilities for all the other states, denoted by p = $\{p_1, p_2, \ldots p_n\}$. The confidence percentage, at any time-slot, is calculated by multiplying the probability of the state at this time-slot with the state probability at the previous time-slot. Hence, at time-slot t+1, the confidence to be at state st_1 is calculated as (p_i*p_1). The confidence calculated should be more than certain threshold to continue the prediction in this direction.

Continuing the previous example, confidence vector CP at time-slot t+1 would look like CP = $\{p_i*p_1, p_i*p_2, \ldots p_i*p_n\}$. At this step, the vector CP is put into a priority queue. The queue sorts the values in CP vector in a way that the highest confidence value is put at the top of the queue. This means, if the value (p_i*p_1) resulted into the highest confidence value, the path prediction continues from state st_1 for next time-slot t+2. The process is repeated for each coming time-slot, until the confidence drops below the threshold value.

The selected paths are all the states selected with confidence greater than the threshold. The confidence reduces as we go further from the start time-slot. In the figure 4.14, the selected paths are i-1-5 and i-1-4. This means the final confidence calculated for path i-1-5, which is $(p_i*p_1*p_5)$ is greater than or equal to the confidence threshold and for

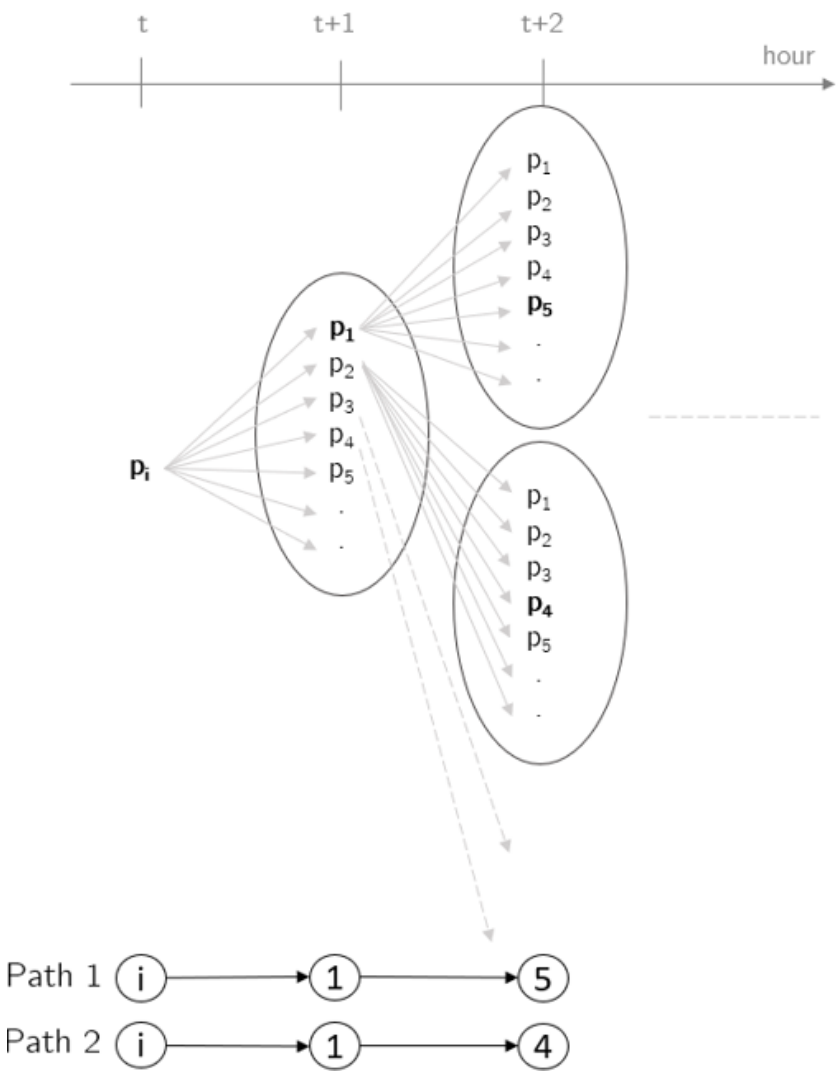

Figure 4.14: Privacy-risk estimation Algorithm using Markov Chain Model

path i-1-4, which is (p_i*p_1*p_4) is also greater than or equal to the confidence threshold. The process continues till the confidence drops below the threshold value. At the end of each predicted path, an additional state is also represented. For this additional state, the confidence will be less than threshold value. This is done to indicate the confidence drop right after the last selected state, in each predicted path.

4.3.2 Proposition: Calibration of Privacy-risk Estimation

The privacy-risk estimation algorithm is to suggest a user his/her predictability. The algorithm reduces the confidence as we go further away from the starting time-slot. This

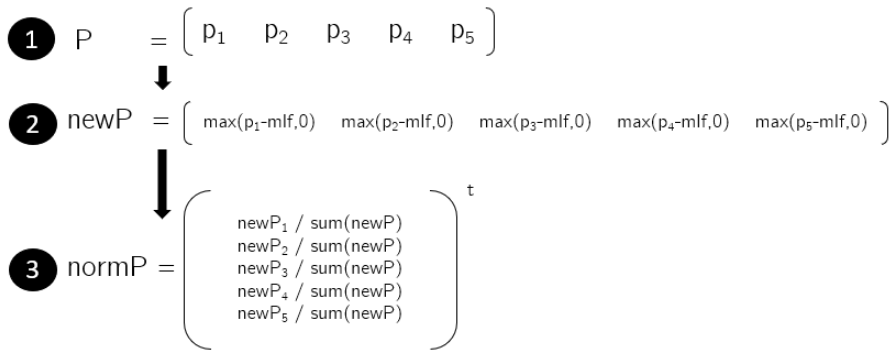

Figure 4.15: Steps to apply memory-loss factor to the Markov Chain

is because of the way markov chain model works. The model keeps a record of each transition.

The humans may have different expectations for their predictability. Most of us ignore the infrequent and minor transitions. For instance, the exact place and time for the lunch 1 week ago is easily forgettable. But markov chain model record each of these transitions. This affects the predictions made during these hours where several infrequent transitions take place. Hence, markov chain model also need to forget these minor transitions to have the same results as expected by the users of the application. This is called as applying memory-loss factor to the markov chain model.

The memory-loss factor mlf can be calculated and applied to the markov chain model. This can be implemented if the probabilities are reduced by a factor and then normalized again. This will strengthen the higher probabilities and diminish the lower probabilities completely. The process is explained with the help of a the figure 4.15 . It is important to remember that the markov chain model contains probabilities of transitioning from one state to another. In step 1, the relevant probability vector is extracted. In step 2, the memory-loss factor mlf is subtracted from each of these probabilities and if the subtraction results into a negative number then it is zeroed. In step 3, the vector is re-normalized by dividing each element with the sum of the vector.

Figure 4.16 depicts an example with memory-loss factor mlf as 0.05. When this factor is applied to the probability vector P, the stronger probabilities like 0.79 is strengthened to 0.91 and all the other probabilities are reduced. The very small transition probabilities like 0.03 are zeroed.

The application of memory-loss factor on markov chain model will make the strong probabilities stronger and the weak probabilities weaker. The privacy-risk estimation algorithm will consider these new probabilities. The so-called confidence percentage,

$$mlf = 0.05$$

$$\begin{array}{lll} \textbf{1} & P & = \begin{pmatrix} 0.79 & 0.12 & 0.03 & 0.03 & 0.03 \end{pmatrix} \\ & \downarrow & \\ \textbf{2} & newP & = \begin{pmatrix} 0.74 & 0.07 & 0 & 0 & 0 \end{pmatrix} \\ & \downarrow & \\ \textbf{3} & normP & = \begin{pmatrix} 0.91 & 0.08642 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 4.16: Example of application of memory-loss factor

which is calculated as the multiplication of the probabilities from one time-slot to the next, will be large for important locations and diminish quickly for infrequent locations. The estimation will result into path predictions closer to human expectations. Hence, this step calibrates the privacy-risk estimation algorithm.

5 State Formation

In this subsection we introduce the of state formation for the prediction model. The sections include the explanation of each component involved in detail and the corresponding algorithms.

5.1 Variables Used

Figure 5.1 is to provide an overview of the variables used in the further sub-sections. The variables are used in algorithms and in explanation. The list of variables covers majority of variables used in further sections, but the list is not exhaustive. Few new variables are introduced and explained in the coming sections for clear understanding of concepts.

Variable	Description
P[x, y, d]	GPS raw coordinate point: (Latitude, Longitude, Datetime)
lst_ hr_ pts[x, y, d]	Last hour GPS coordinate points (Latitude, Longitude, Datetime)
sp	Stay-point
st	State
w	State hour weights
mc	Markov Chain
th_ tck	Threshold time for tracking GPS location data
th_ d	Distance threshold
th_ t	Time threshold

Figure 5.1: List of Variables

5.2 Stay-points Detection

The first step for state formation is stay-point detection. Stay-points are those important places where user spends ample amount of time. Distance and time-based clustering work best in case of location data as explained by the authors [7]. This clustering approach is not so complex and can be run on a mobile device as a background process. The clustering has two thresholds, one for distance th_d and one for time th_t. These thresholds help us to determine the stay-points in an online fashion. The location points within the radius of distance threshold th_d and time spent at this location is greater than or equal to the time threshold th_t, are together regarded as a stay-point. For example, a set of points within 50m of radius and total duration of stay greater than 10 minutes, can be regarded as a stay-point. In the example above, the café location will be a stay-point only if the stay is greater than or equal to the time threshold th_t. This help to remove noise, like travelling location coordinates or short stay locations. Hence, only significant locations from the trajectory are extracted and noise is removed.

5.2.1 Sporadic GPS Trajectories

A GPS trajectory or location trajectory is the path taken by the user where the GPS points are continuously received i.e. every 5-10 seconds a coordinate is received. A trajectory can end for several reasons. For example, if the user turns-off the phone or turns-off location sharing, or the user enters a low network area. The clustering approach works adequately if the GPS points are received continuously. But if the location information is collected only at intermittent intervals, the extraction of all the important locations is incomplete. This type of GPS trajectories contributes to the sporadic GPS trajectories.

Stay-points are any points which are stayed by the user during the user GPS trajectories or it is the start or the end of the GPS trajectory. For example, if the user starts his/her trajectory at “home” location, the “home” itself is a stay-point. Now he moves towards “work”, but he visits a “café” in between for breakfast. The “café” is also, a stay-point and then he finishes his trajectory at “work”, where “work” is again a stay-point.

Location Trajectories are continuously received location coordinate points. The time difference greater than tracking time threshold th_tck between two location coordinates breaks the old trajectory and starts a new one. This means, if the location coordinates are received continuously for few hours, then the location coordinates are stopped, therefore ending this trajectory. As soon as the new location coordinate starts, a new trajectory will start. Note that the stay-points, within the trajectory, are found using the time and distance clustering algorithm [7]. But, the location coordinate points

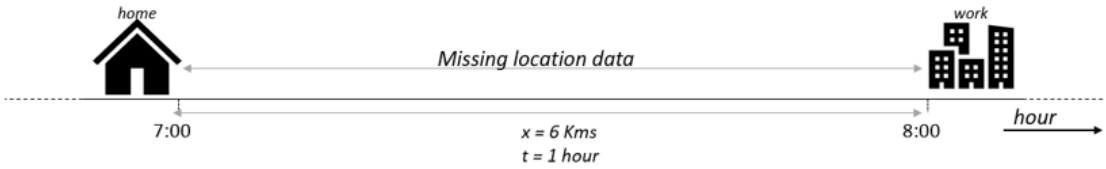

Figure 5.2: Example 1 of sporadic GPS trajectories

where user has ended or started his/her trajectory, are also considered for stay-point detection.

5.2.2 Estimating Arrival and Departure times of Stay-points

After the collection of stay-points, the stay-points entering and leaving time is recalculated. This is done to estimate the time of leaving a stay-point and the time of entering the next stay-point.

Case 1:

Let us understand this using an example as depicted in figure 5.2. In this example, we consider the distance threshold th_d to be 200 meters or 0.2 kms. For instance, user is reported to be at “home” location at 7am and then the next stay-point is found to be “work” location at 8 am. The time of departure from “home” location and the time of arrival at “work” location can be estimated.

Furthermore, in figure 5.2, the distance between these two locations is x kms, which is easy to calculate as the “home” location coordinates and “work” location coordinates are known. Let’s consider the distance x to be 6 kms between “home” and “work” location. The time difference t between the two points “home” and “work” is also known, which is 1 hour in this example. This information helps us to estimate the actual leaving time from “home” location and actual arriving time at “work” location. The speed of user spd can be calculated as (x kms / t minutes) i.e. (6/60) kms/mins or 0.1 kms/mins. Now, the delta time delta_t is calculated as minimum(th_d, x)/spd i.e. minimum(0.2, 6)/0.1 or 1 minute. The delta_t is added in the departure time of “home” location and subtracted from the arrival time of “work” location. So, the estimated departure time at “home” location is 7am + delta_t i.e. 7am + 1 minute and the estimated arrival time at “work” locations is 8am – delta_t i.e. 8am - 1 minute.

Case 2:

Although, there could be the case where the user is not travelling or moving from one location to another, but rather the user stays at a location after some missing data.

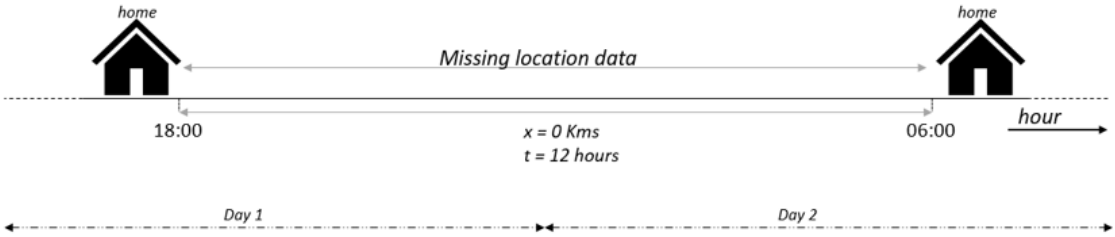

Figure 5.3: Example 2 of sporadic GPS trajectories

Consider the example shown in figure 5.3 where user is at "home" location at 18:00 and now the location data is not shared for some reason for the next few hours. The next location shared is again "home" location at 06:00 the next day. The missing location data is most likely the "home" location for the entire time-slots between 18:00 on this day till 6:00 on the next day. Since the distance between these two points is 0, the speed spd of travel will also results in 0 km/hour. Now, the estimated time of being at "home" location is recalculated. The time difference between the two known points is 12 hours. The time of leaving "home" location is recalculated as (18:00 + 12/2) i.e. at 00:00 on this day and the time of arriving at "home" location for the next day is (06:00 – 12/2) i.e. at 00:00 on the next day. In other words, it is estimated that the user stayed at "home" location for this period.

5.2.3 Algorithm

The stay-points are extracted from raw points to remove the noisy points. The noisy points could be travelling with the bus or train or a short stop at the letter box. The stay-point extraction is the process of extracting longer stayed locations from raw GPS trajectories. The figure 5.4 shows the transition from "home" to "work". In this case, both "home" and "work" are extracted as stay-points. The two bigger circles denote the stay-points and the smaller dots within represents the location coordinate points. The radius of the stay-point circles is the distance threshold th_d.

The extraction of stay-points takes lst_hr_pts as input and generates stay-points sp = $\{sp_1, sp_2, \ldots\ sp_n\}$ as output. The algorithm cluster the points within the radius of stay-point distance threshold th_d for a minimum duration of time th_t. The selection of distance and time threshold is very important. If the distance threshold value is too large, the mean of the stay-point locations will be a confusing location on the map and many unimportant locations will become part stay-points. If the time threshold value is very small, a lot of insignificant locations with short stay duration will be added as

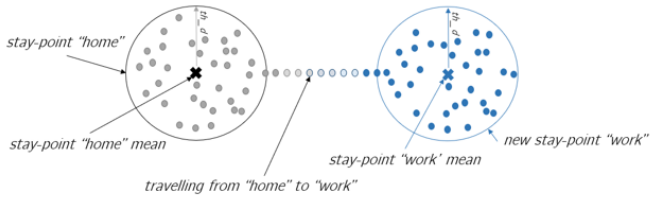

Figure 5.4: Extracting stay-points from GPS Trajectories

stay-points. A balanced selection of distance and time threshold can range from 100-300 meters and 10-30 minutes respectively.

Algorithm 5.1 explains the approach used for stay-point extraction. The GPS location points P[x, y, d] are received and stored in last hour points lst_hr_pts until the next hour. The points from lst_hr_pts is read and the points which are close-by are added to the same cluster. A new location from lst_hr_pts is added to the cluster if the distance between the new point and the cluster mean is less than or equal to the distance threshold th_d. Every time a new point is added to the cluster, a new mean of the cluster is calculated and the process repeats. If the new point is moving away from the cluster mean, then the point is not added to the same cluster. This means, that if the distance between the mean of the cluster and the new point is greater than threshold th_d, then the new point is not added to the same cluster. At this point, this cluster duration is checked. The cluster duration is nothing but the largest date-time – smallest date-time from the cluster elements. If the cluster duration is greater than or equal to th_t, then the cluster is added as the stay-point sp with latitude and longitude as cluster mean. Otherwise the cluster is not added as a stay-point and the cluster is removed.

A new point is also considered for stay-point if the difference of time between the new point and the previous point is greater than time tracking threshold th_tck. This is to ensure that if the location coordinate points are not received for a long time then, end of the previous trajectory and the start of the new trajectory points are also considered.

The stay-points are often not continuously distributed over time. Consider a scenario where a user is at "work" location till 9 am. After entering "work", there are no points received for next few hours. The next stay-point detected is "work" location at 1 pm when user came out for lunch from the building. The time of stay at "work" in this case till 9 am is misleading as user stayed at this location till 1 pm. Hence, once the stay-points are collected, we adjust the starting time and the leaving time of each stay-point.

The algorithm 5.2 details how to achieve this. A comparison of each stay-point in sp $= \{sp_1, sp_2, \ldots\ sp_n\}$ to it's very next stay-point is done. Now, the distance d and time

Algorithm 5.1 extractStayPoints() : Stay-point Detection

Input: Last hour GPS points *lst_hr_pts[x, y, d]*
Output: Stay-points *sp*

```
for each lst_hr_pts[x, y, d]_i in lst_hr_pts[x, y, d] do
    if d_i − d_i+1 > th_tck then
        sp ← lst_hr_pts_i, lst_hr_pts_i+1
    else if distanceBtw(lst_hr_pts_i, cluster) <= th_d then
        cluster ← lst_hr_pts_i
        Update Cluster Mean
    else if (cluster! = empty) And duration(cluster) >= th_t then
        sp ← cluster
    end if
end for
```

Algorithm 5.2 adjustStartEndStaypoint() : Adjust start-end time of stay-points

Input: Stay-points *sp*
Output: Stay-points with adjusted start and end points *sp*

```
for each sp_i in sp do
    d ← distanceBtw(sp_i, sp_i+1)
    t ← timeDiff(sp_i, sp_i+1)
    s ← d/t
    if s! = 0 then
        delta_t ← min(d, th_d)/s
    else
        delta_t ← t/2
    end if
    Update end_time d_e+delta_t for sp_i
    Update start_time d_s-delta_t for sp_i+1
end for
```

difference t between the two stay-points sp_i and sp_{i+1} is calculated. Using distance d and time t, the average speed s of travel can be easily calculated, which is d/t. If the speed s is not 0, the delta time delta_t is calculated as division of minimum of distance between sp_i and sp_{i+1} i.e. d and threshold distance th_d, to the average speed s. If the speed s is 0, this means that the user has not displaced from the previous location and hence, the time difference t is filled with the same location. Now we add the delta time delta_t to sp_i end time, to change the leaving time at the sp_i location and subtract delta_t for sp_{i+1} start time to change the entering time at location sp_{i+1}.

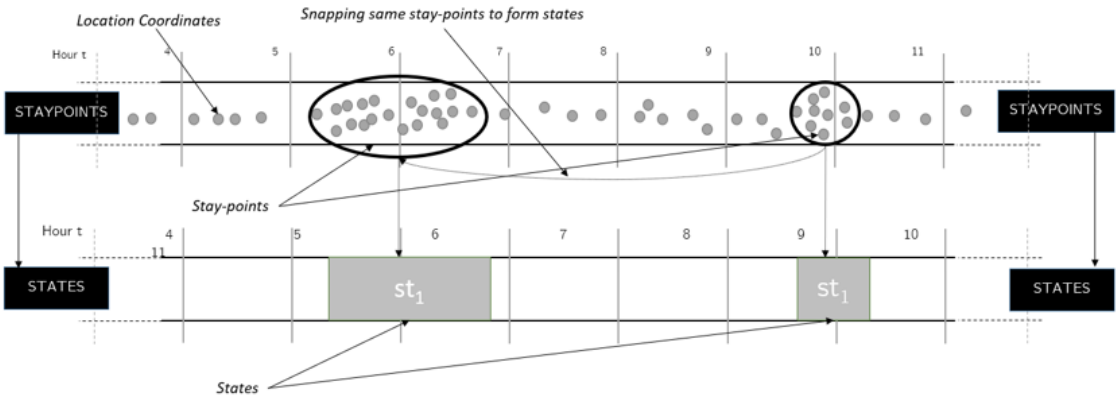

Figure 5.5: Snapping stay-points to states

5.3 State Formation from Stay-points

From the stay-points sp, the states st = $\{st_1, st_2, \ldots\ st_n\}$ are formed. This process snaps the geographically close-by stay-points to one state. Similarly, several states are formed from stay-points. The number of states can be less than or equal to the number of stay-points. These states are used later for markov chain model. The states' st represent "home", "work" and other important visited places. This means, the same locations visited are snapped to the same state ids. For example, user has been at location "home" during the early hours of the day. If location "home" is visited again during the day, it will be extracted as a new stay-point. But, since "home" location was visited already known, it will be snapped to an existing state.

The states are found using a distance threshold th_d. All the stay-points within this threshold distance radius is grouped together as a single state. This is called snapping stay-points to the states. This is depicted in the figure 5.5. The two stay-points are snapped to states to form st_1. The first stay-point between hour 5 and 7 is assigned to a state st_1. When the second stay-point between hour 9 and 11 is extracted, the distance between the stay-point and the state st_1 is checked. Since the distance between the states is less than the defined threshold distance th_d, this new stay-point is also snapped to the same state st_1. This makes sure that if a known location is visited after few days, then it should get the same id as given before. Imagine a case where the user visited a restaurant last Monday and visits the same restaurant again the next Monday. In this case, the restaurant visit on next Monday should get snapped to the already existing state from last Monday. The markov Chain model is applied to these states.

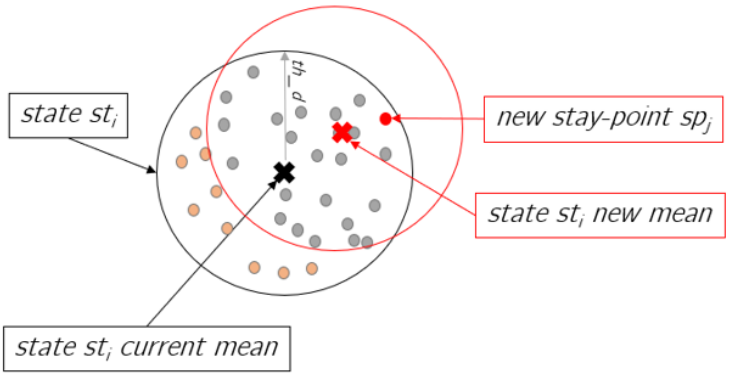

Figure 5.6: Drifting problem while snapping stay-points to the states

5.3.1 Drifting Problem

The stay-points are snapped to an existing state with an exception. The figure 5.6 depicts the drifting problem. The current mean of state st_i is marked with black $\times$. A new stay-point marked with a red dot on the right has arrived. The addition of the new stay-point sp_j to this state will make the mean of the state shifted, denoted by red $\times$. The new mean of the state st_i will throw some of the existing points from left out of the state radius. Therefore, the new stay-point sp_j is not added to a the state st_i. Hence, the idea is, while adding the new stay-point to an existing state, a check is done. If all the existing stay-points stays within the radius of the state mean, then the stay-point is snapped to the state, otherwise a new state is formed. So, if any of the existing stay-points, contributing to the state formation, is moving out of the state radius, then the new stay-point is not added to this state to avoid the drifting problem.

5.3.2 Algorithm

Algorithm 5.3 explains the details of state formation from stay-points. Each new stay-point goes through this step. When a new stay-point sp_{n+1} is detected and added to the stay-points $sp = \{sp_1, sp_2, \ldots\ sp_n, sp_{n+1}\}$, the stay-point sp_{n+1} distance is compared with all the existing states in $st = \{st_1, st_2, \ldots st_n\}$. If the distance between a state and the new stay-point sp_{n+1} is less than the distance threshold th_d, then the stay-point sp_{n+1} added to the that state. Otherwise, a new state is formed.

In the algorithm, the states are represented with a numeric id number. The process ensures that we keep snapping the known locations with the same ids. Once the day

Algorithm 5.3 formStates() : Form states from stay-points

Input: Stay-point sp_{n+1} , States st
Output: States st

```
for each st_j in st do
    is_sp_{n+1}_added ← False
    if distanceBtw(sp_{n+1}, st_j) <= th_d then
        Add sp_{n+1} to st_j
        newMean ← mean(st_j)
        if distanceBtw(All sp's of st_j, newMean) <= th_d then
            is_sp_{n+1}_added ← True
            Break
        else
            Remove sp_{n+1} from st_j
        end if
    end if
end for
if is_sp_{n+1}_added = False then
    Add new state in st
end if
```

is changed, the markov model mc is created. The algorithms of markov chain model creation is explained in previous chapters.

As the number of states increase, the complexity of the markov chain model also increase. Let us consider an example of markov chain with states. The figure 5.7, on the left, depicts two states st_1 and st_2 markov model. This model involves 4 probabilities p_1 to p_4. Each of these probabilities represents the transition probabilities from one state to another. For instance, p_1 is the transition probability from st_1 and st_2. Figure 5.7 on the right, depicts a third state added to the model st_3, which in turn, increase the probability count from 4 to 9. This explains, how the complexity and the computation of the model increase as the number of states increases in the model.

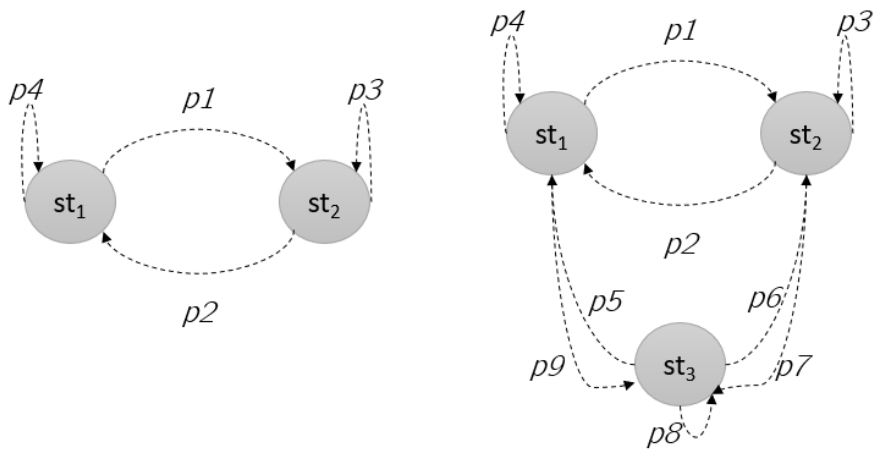

Figure 5.7: Markov chain on states

6 The PRE Android Application

The implementation is done on an android phone as a prototype. This is to showcase how the mobile devices receiving location data can exploit the data and make predictions. The data comes from Geolife dataset. In the actual scenario, this data will be fed from GPS to the application. The android implementation has challenges like computational cost. The implementation is similar as the python implementation. The user can see the prediction results as path with corresponding confidence levels.

6.1 Objective

The intention is to have an operational android application which can make path predictions and showcase the privacy threats of location-based services. The locations predicted in each step should be within a certain confidence percentage. Visualization of the predicted paths should be understandable to the user.

6.2 User Interface Design

The user interface comprises of 3 sections. Each section consists of an independent screen for user interface.

Screen 1: The start screen of the android application, as shown in the figure 6.1, has two modes “GPS” or “Geolife User Data”. The option “Geolife User Data” is available to use the Geolife user dataset. The drop down lists of choosing user and month is available when "Geolife User Data" is chosen. The user and month must be selected from this input screen to be able to go forward. The “GPS” mode will be used in the actual application usage scenario where the user location details will be read for few weeks.

Screen 2: Once the user has made the selection and hit “Continue” button on screen 1, the markov chain is built in the background from the user data file. The states are extracted which represents the significant places for the user. These states are displayed as shown on left of the figure 6.2. The states may not be the exact positions user visited,

but rather in the vicinity, as these states represent the mean of several coordinates within a range. The list is displayed to show the user all the visited placed that has been tracked down as important places.

Screen 3: Once the user proceeds by pressing the button “Find Predictions”, user is taken to the final prediction screen. Here the user can select from the list of states and hour from the screen. The state selection is user’s known location and the hour selection is the time-slot selection. The markov chain can use this selection to predict his future visits. The confidence threshold input can be changed using the slider. This threshold controls till when the paths are predicted. The confidence threshold ranges from 10% to 100%. A confidence threshold as low as 10% can generate many longer paths and a higher confidence threshold as 90% can produce fewer shorter paths. Once the selection is done, the button “Run” can be hit to produce the predicted paths. The paths are displayed below as shown in the figure 6.2 on the right. The paths are displayed with the help of states. The states confidence is represented by the help of the font color. The darker state font colors depict higher confidence and vice-versa. In the example figure 6.2 on the right, two paths are predicted. The start hour and end hour are shown in parentheses, at the origin and the closing of the path respectively. The example below shows the known state is 3 at hour 0. The predicted paths start from hour 1 till hour 15. Path 1 suggests that user will transition from state 3 to state 6 and will stay at state 6 till hour 15. Whereas path 2 suggests that user will stay at state 3 for longer duration. It is also easy to spot that the prediction confidence of path 2 drops earlier (as the color starts to fade away quite earlier) than that of path 1. Hence, the prediction output path 1 is more confident than prediction path 2.

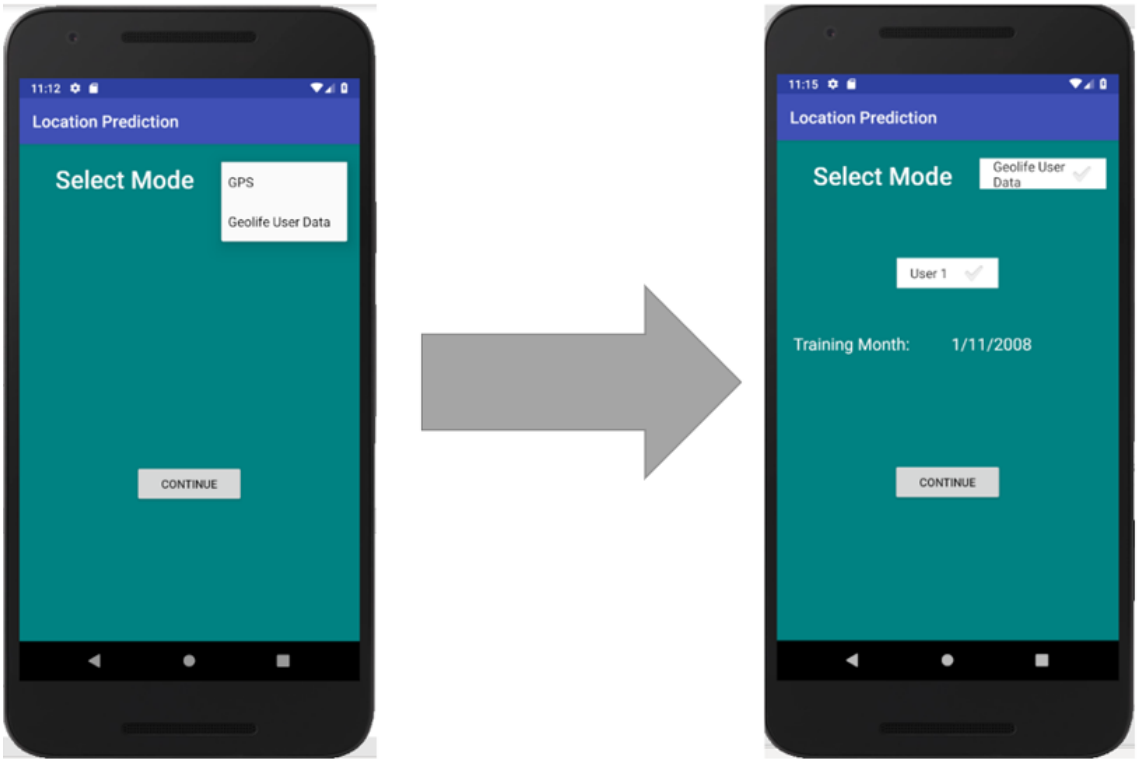

Figure 6.1: Android application screen 1

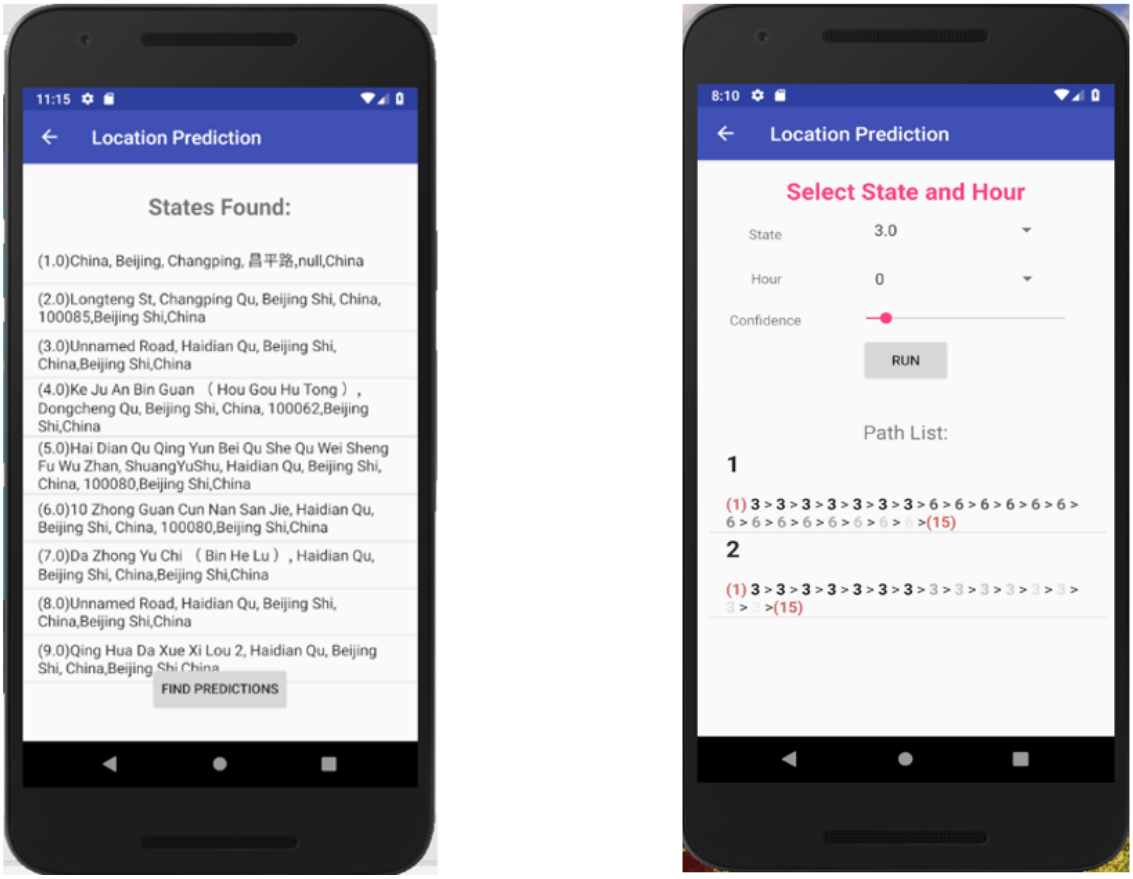

Figure 6.2: Android application screen 2(left) and 3(right)

7 Evaluation

In this chapter, we evaluate the algorithms explained in the above sections on Geolife dataset. We also evaluate user datasets to extract the meaningful data and then the prediction results by the application of the algorithm.

7.1 The Dataset and its Analysis

The data used for this master thesis is Geolife dataset from Microsoft [17] [15] [16]. The dataset contains 182 users' GPS trajectory data for the period of five years. The trajectory data contains the latitude, longitude, date, time, altitude information which is tracked every 1 to 5 seconds. Majority of the dataset was created in China with few exceptions of USA and Europe. This included several different types of users, few with a lot of trajectory data over years and few only for few weeks. For instance, user 17 has 1026179 trajectory points and user 72 has only 81.

A total of 73 users have also labelled their transportation mode while recording the GPS trajectories. The figure 7.1 explains the distance travelled altogether using different transportation modes.

The transportation labels summary helped us to understand that most users were travelling while recording their GPS trajectories. The average trajectory speed, which is calculated per trajectory file for each user in the dataset, is depicted in the box plot in figure 7.2. The median travel speed was found to be 5.73 km/h. This indicate that most users tracked the GPS trajectories outdoor.

The analysis of each user also shows some important patterns. For instance, the first and last GPS coordinates of each trajectory for user 1 is depicted in the figure 7.3. The green dots indicating the start of each trajectory and the red dots indicting the end of each trajectory. This also shows a pattern indicating that the trajectory data was mostly recorded outdoor. For example, the trajectory was starting at home and ending at work and repeated in cycles. This analysis motivated us adding the starting and the ending point of each trajectory as stay-points. The approach is explained further in the chapters before.

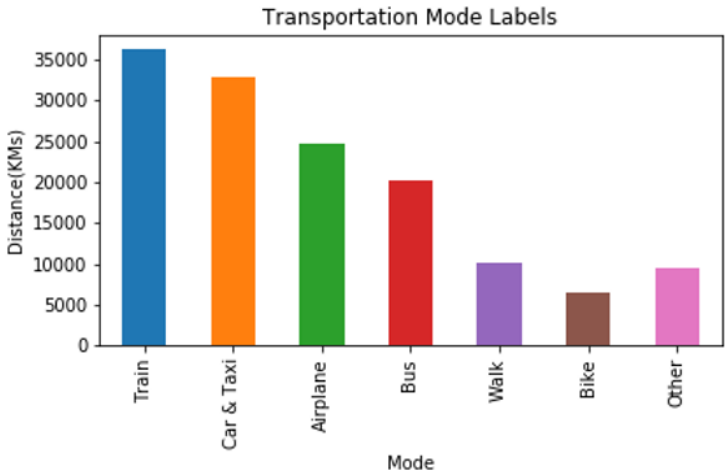

Figure 7.1: Geolife dataset user transportation mode

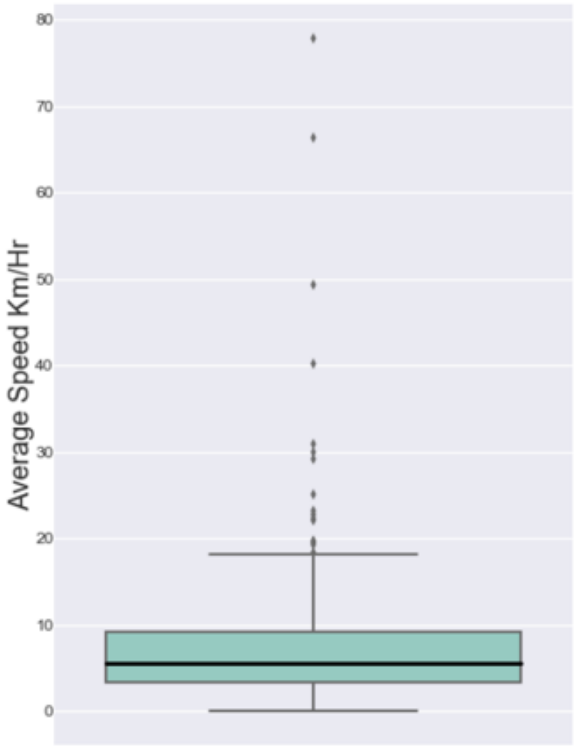

Figure 7.2: Geolife dataset trajectory average speed

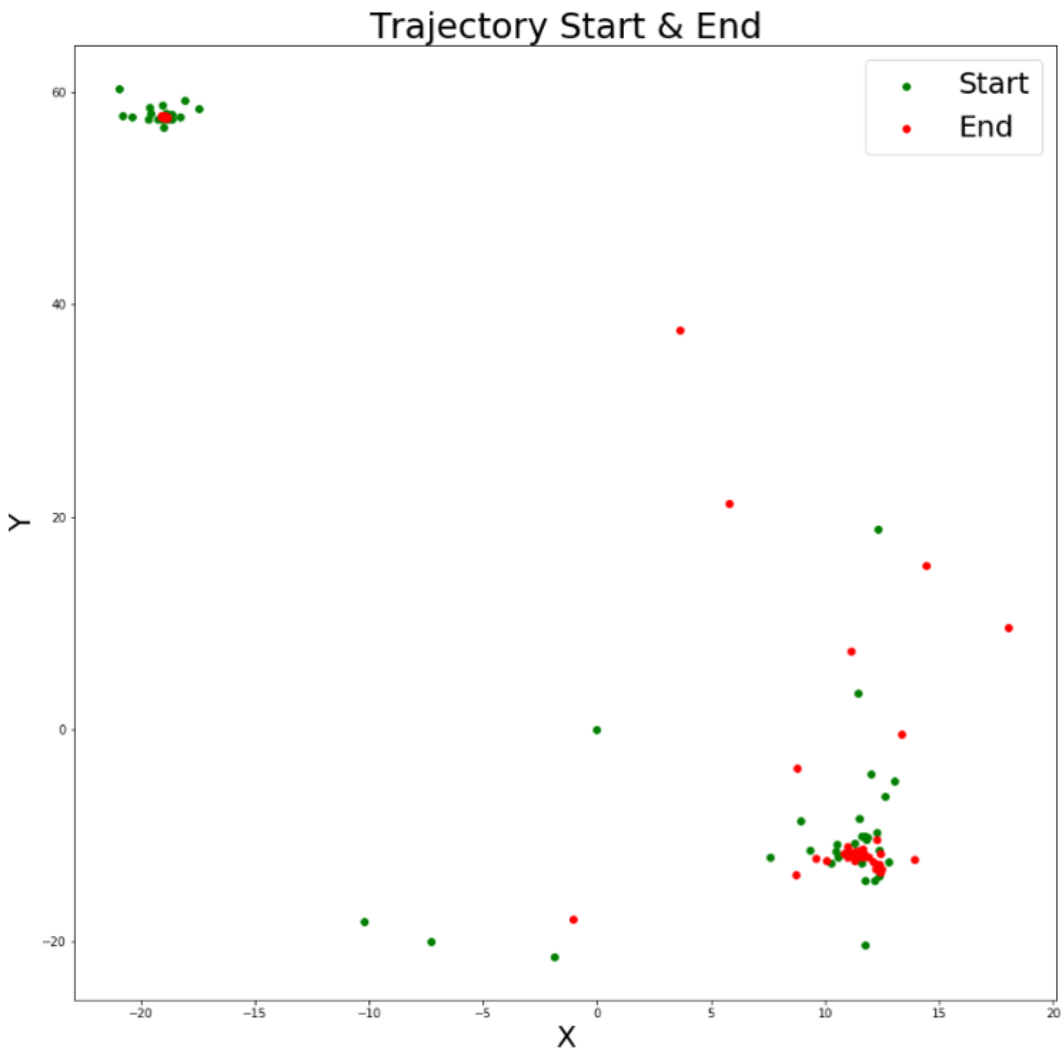

Figure 7.3: start and end points from each trajectory

7.2 Evaluation and Results

In this section, we detail evaluate each individual component of the algorithm.

7.2.1 Stay-points Evaluation

The algorithm is applied to Geolife dataset. The user files are read in an online manner to simulate the GPS location points received on a mobile device. The stay-points distance and time threshold values are important to be appropriate. If the distance threshold

is very large, a lot of insignificant locations will be part of stay-point cluster and if the time threshold is very small, the short duration stays will also be regarded as stay-points. The figure 7.4 shows the stay-point count on a user trajectory trace for varying the distance and time threshold values in range 50m-350m and 10min-30min respectively. It is evident from the figure that the number of stay-points are very high with small time threshold and reduces as the time threshold increases. The time threshold value at both the extremes results either in very less stay-points or many stay-points, hence we select a value from the middle. Performing several tests analysis, we chose 200 meters and 20 minutes for distance and time threshold respectively. This threshold values can vary with the type of data.

The stay-points found for user 1 for November 2008 as shown in the figure 7.5. The trajectory is shown with the green line and the red arrows indicate the stay-points. These stay-points represent the locations with semantic meaning behind it like “home”, “work”, or “sports club”. The green line has a lot of unimportant locations and short stays. These locations are ignored for stay-point extraction and only the significant red arrows are marked as important locations.

The stay-points algorithm considers the start and end trajectory points and smoothens the data between two stay-points, based on the distance and time elapsed between them. The result of this approach is compared against the stay-point extraction within the trajectory points. The results are shown in figure 7.6. The data used is from Geolife dataset for user 1 for the first few days in November. The x-axis shows the time-slots and the y-axis shows the days. The top of the figure 7.6 shows the black marked rectangles which represents the raw trajectory data. The figure indicates that a lot of time-slots are missing data, especially the data after hour 23 till hour 7 in the morning. The middle portion of the figure indicates the stay-points found only within the trajectory points. The time-slotted data here does not represent any regular pattern and misses the important locations. The bottom of the figure depicts the stay-points found using our approach. The pattern “home-work-home” is now very evident. Hence, this approach does not miss the most important states like “home” and “work”.

7.2.2 Time-slotted Data

The hourly weights associated with the states forms the time-slotted data which is used as the base for markov chain model.

It is inferred that the occurrences of a few popular locations like “home” and “work” for a user will be more, compared to other locations. It is very often that the user stays at “home” location after the evening and spends more time at “work” location during the day. Of course, there could be night-shifts, but then the duration of stay at “work”

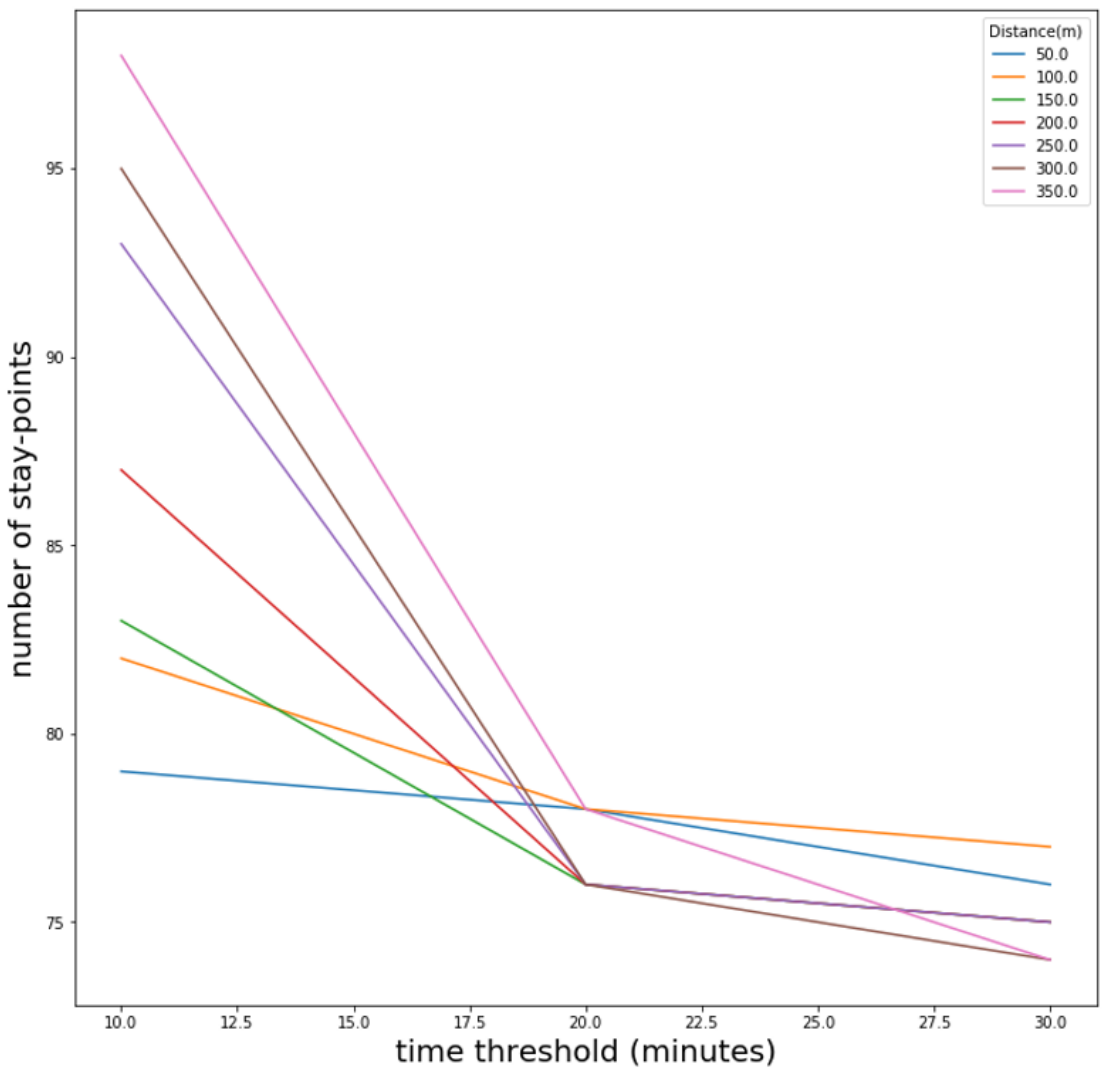

Figure 7.4: Stay-points count for different time and distance thresholds

location, which usually ranges from 8-9 hours, can help to make the right indications. The "home" location is also often the one which has occurrences during weekends or public holidays. These indicators help us in marking the "home" and "work" location while analyzing the data.

The Geolife dataset date and time are represented in GMT, hence to have the correct visualization, the date and time must be adjusted to the local time in the trajectory data. The figure 7.7 below depicts the hourly weighted state data for user 1 for November 2008. The x-axis represents each hour of the day from 0 to 24 and the y-axis represent the days. Each rectangle represents a state where the width of the rectangle represents the weight of the state in the corresponding hour. For example, state 1 is the first state

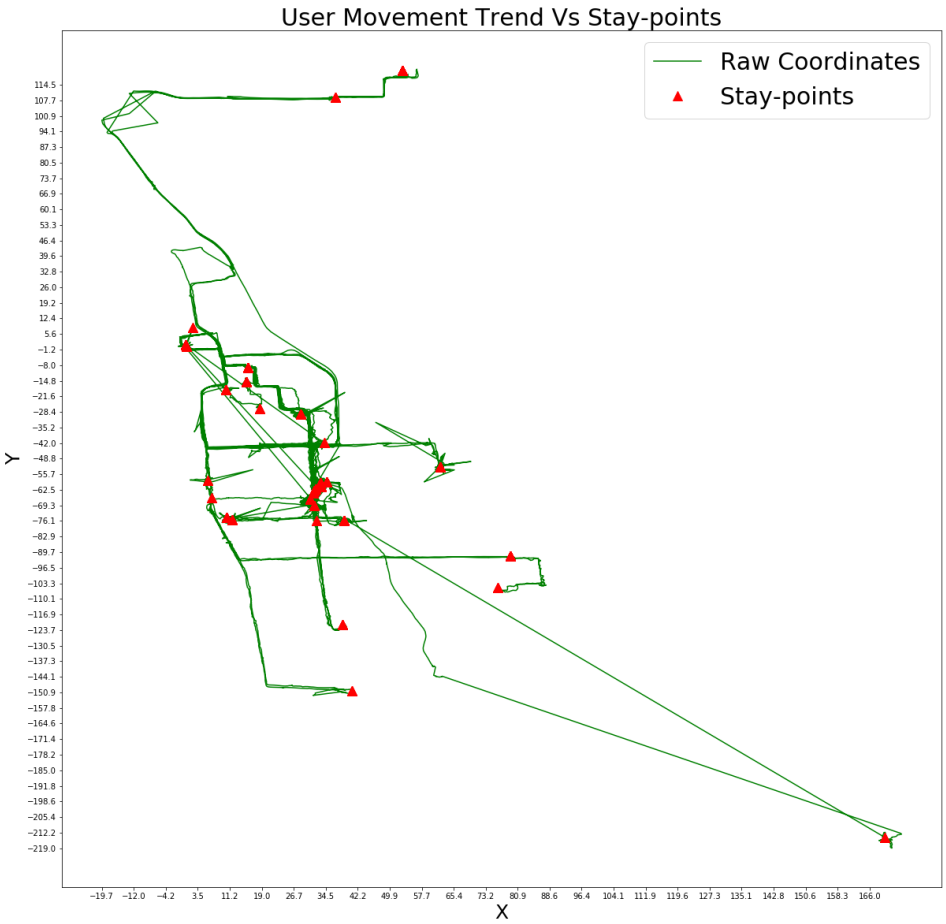

Figure 7.5: User 1 raw trajectory data vs. stay-points extracted

between 9 am and 10 am on day 0. The distribution of the states over the hours and days makes some hints about the semantic meaning behind the locations. For instance, state 3 is most likely user's home location as on most days' user is at this location from 9 pm till next day 7 am and state 6 is most likely user's work location as on most days' user is at this location from 8 am till 8 or 9 pm. There are many other locations like 13, 22, 2, 16 and so on which could represent the supermarket, shopping mall, fitness club and so on.

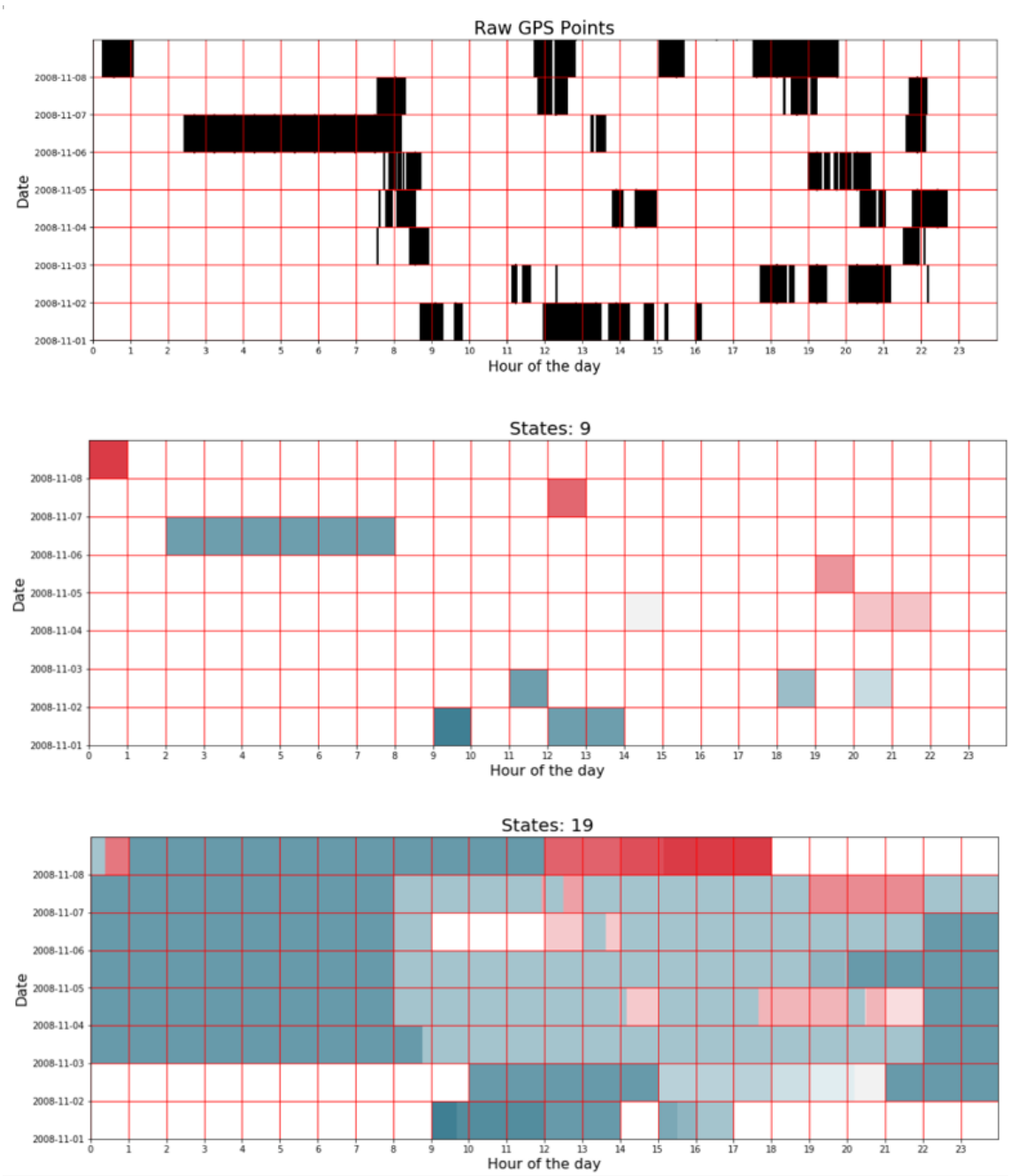

Figure 7.6: User 1 eight days' time-slotted; 1. Top: Raw Trajectory Data, 2. Middle: Stay-points within trajectories only, 3. Bottom: Stay-points with our approach

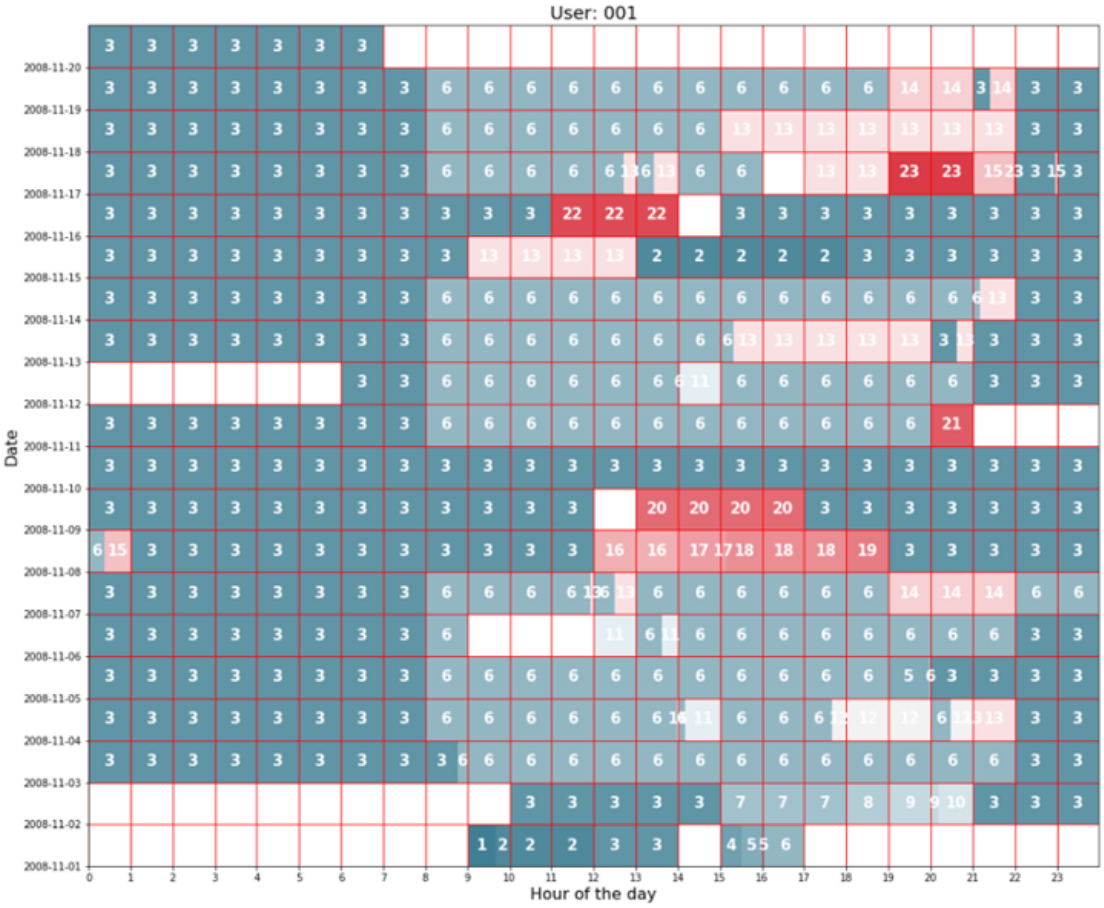

Figure 7.7: User 1 time-slotted data for the November 2008

7.2.3 Privacy-risk Estimation Evaluation

The algorithm takes the current hour, location and minimum threshold as input and uses the markov model to predict the several paths representing the several locations in consecutive time-slots with their confidence percentages. The figure 7.8 shows the paths predicted for user 1 for a known state 3 at hour 7 with minimum confidence 0.1. The x-axis denotes the hour of the day and each path starts in a new line. There exist two paths. The circle with the number denotes the state predicted and the color of each circle is to indicate the confidence of the prediction. The first predicted state at hour 8 in path 1 is 6. Similarly, the first predicted states at hour 8 in path 2 is state 3. The states with darker color indicate a high confidence and the states with lighter color indicates the lower confidence. The confidence reduces as we go further away from the starting hour i.e. 7. The path continues until the confidence drops below the threshold confidence. There is an additional state shown after the drop of confidence

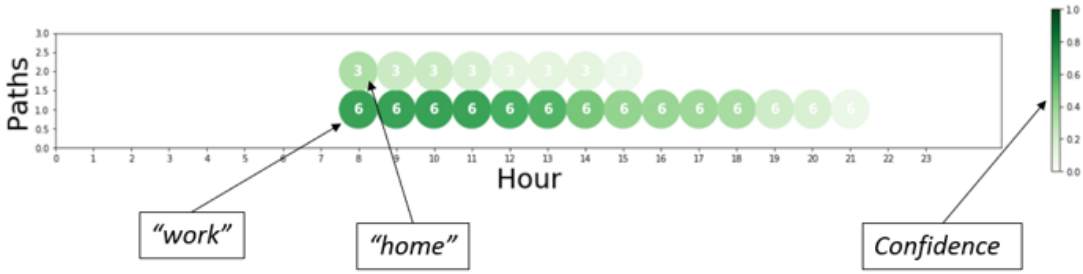

Figure 7.8: Path prediction for user 1

below threshold. This additional state is shown to see the drop in the confidence of the next state after the minimum confidence.

The privacy-risk estimation is done to aware the user about the different paths which are predictable based on his/her location history. The known location was only state 3 at hour 7, which is used to predict the paths for several hours.

7.2.3.1 Survey for Memory-loss Factor

The predicted paths follow a very strict markov chain model. The model is counting each transition. The users of the application would not expect such strict behavior and tend to forget many previous visits. As proposed in the earlier chapters, we can apply this behavior to our model using a memory-loss factor. The memory-loss factor can be calculated using the feedback from the users directly.

Consider the example of user 1 data for November 2011 for first few days as depicted in figure 7.9. The x-axis denotes the hour of the day and the y-axis denotes the days. During the evening the user has been visiting many new places. The transition from hour 18 and 19, as marked in the figure 7.9, is interesting to focus. The state 6 seems to be user's "work" location. The user has visited location 5 and 14 on day 5 and 7 respectively at hour 19, after state 6 "work" location at hour 18. On other days like 3 and 6, he stays at "work" location 6 at hour 19. The markov chain will remember these transitions. This in turn reduces the probability of user to be at state 6 "work" location at hour 19.

As a result of this, the path predicted if the known location is at state 6 "work" at hour 14 is shown in figure 7.10. Because of several possibilities at hour 19, the confidence drops instantly.

But humans do not consider each detail while thinking about the transitions. It is easy to forget the places we have visited 2 weeks before. To improve the path prediction

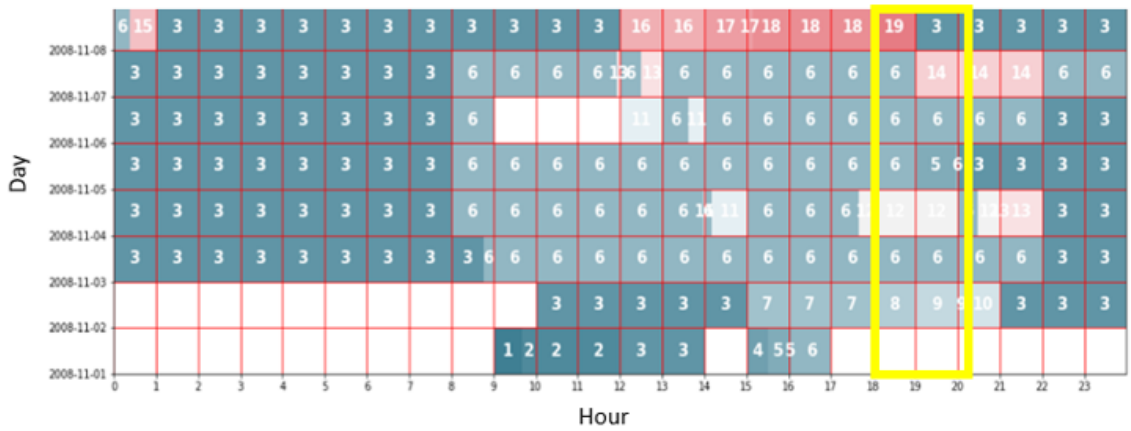

Figure 7.9: User 1 November 2011 hourly distributed data

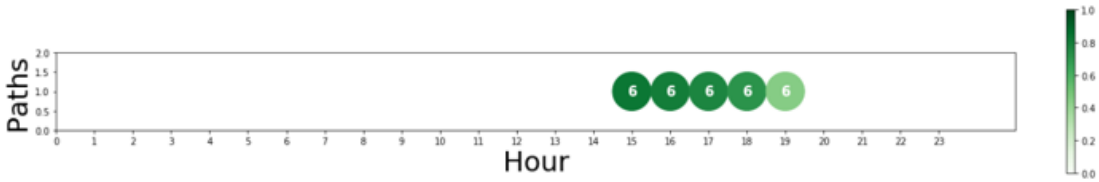

Figure 7.10: Path prediction user 1 with minimum confidence of 50%

algorithm, it is interesting to forget some minor transitions like humans do. This can be done using the inputs from the user, hence, a small survey is conducted.

Survey:

The survey contained 10 participants with the median age of 26.5 years. The participants are shown the time-slotted data as in Figure 33 for 1 minute of time. Before showing the time-slotted data, the participants are explained that the x-axis represents the hour of the day and the y-axis represents several days the data is collected for. Each rectangle represents a location with the id inside the rectangle. They are also told to focus on user's working hours and movement trends. After this, all participants are asked a same question.

"How long will the user stay at location "Work" if he was observed at location "Work" at x hour."

The variable x is replaced with an actual hour value based on the data. The question is to understand how the participants forget the minor transitions and expects the user to be at "work" for longer hours. One of the Geolife user data is shown to the participants which is the figure 7.7. The x in the question was 8am. The average answer from

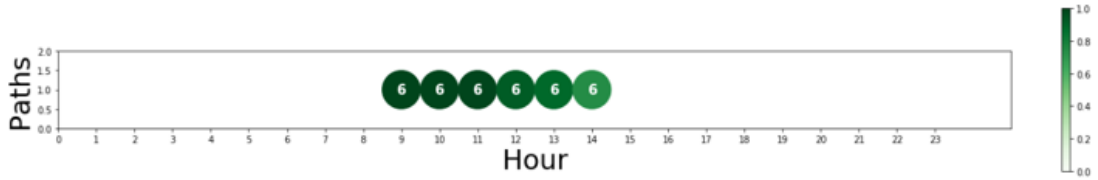

Figure 7.11: Path prediction result for user 1 without changes in the algorithm

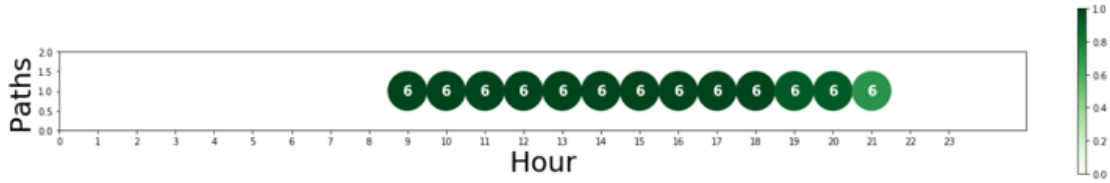

Figure 7.12: Path prediction result for user 1 with memory-loss factor of 0.22

participants reported that the user will stay till 8:00pm at state 6 “work” location. The path prediction from our algorithm resulted in figure 7.11. The prediction is very strict which says user will stay at state 6 “work” only till 14 hours. The minimum confidence considered here is 80% and the prediction of state 6 at hour 14 is already below the minimum confidence. The way participants have observed the movements and the how markov chain has calculated the transitions, are not same. Most participants ignored the infrequent and minor transitions like from state 6 to state 11 from hour 13 to 14 on day 3, and so on. But markov model recorded these transitions and this in turn reduced the probability of staying at state 6 at hour 14. Hence, markov model also need to forget these minor transitions. This is called as applying memory-loss factor to the markov model.

The memory-loss factor is calculated for each participant. The factor is simply calculated by changing the markov model to produce the same output as the participant has suggested. For example, if the participant suggested that the user will stay at state 6 till 9pm, the same output should be produced with our path prediction. This can be implemented if the probabilities are reduced by a factor and then normalized again.

After application of a memory-loss factor, we get the following result as shown in the figure 7.12. The memory-loss factor of 0.22 changes the path predicted and suggests the stay at state 6 till 21 hours.

The figure 7.13 depicts the memory-loss factor from the survey performed for two different data with 10 participants. The median memory-loss factor calculated is 0.17.

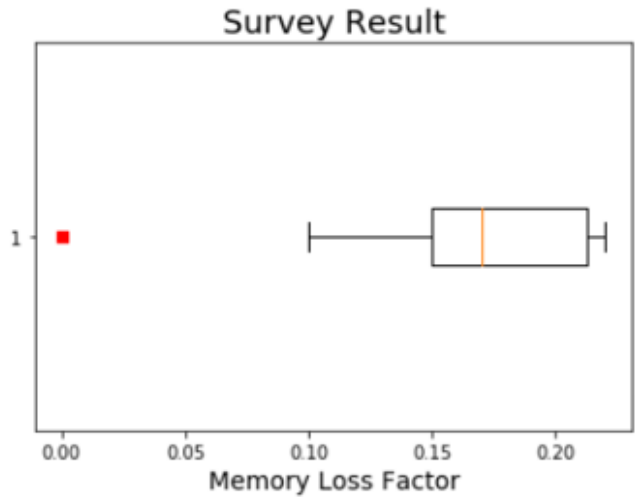

Figure 7.13: Memory-loss factor from survey result

The survey was only conducted with 10 participants. The results can change with more participants and with different age groups. More research must be done in this direction.

7.2.4 Prediction Performance

The location prediction algorithm implemented on Geolife dataset for 182 users. The data is divided for training and test purposes. The user month, with most trajectory points, is selected as the training month and the closest month available to the training month is selected as the test month. This is done to build the model for the month where highest data is available. The closest available month, next to training month, is selected as the test month as this is most likely to have the same pattern as the training month.

7.2.4.1 Metric

The prediction algorithm is evaluated for three parameters, accuracy correctness A_c, accuracy precision A_p and cosine similarity C.

The accuracy correctness A_c is the defined as the ratio of total correct predictions P_c to the total prediction P_t:

$$A_c = P_c/P_t \tag{7.1}$$

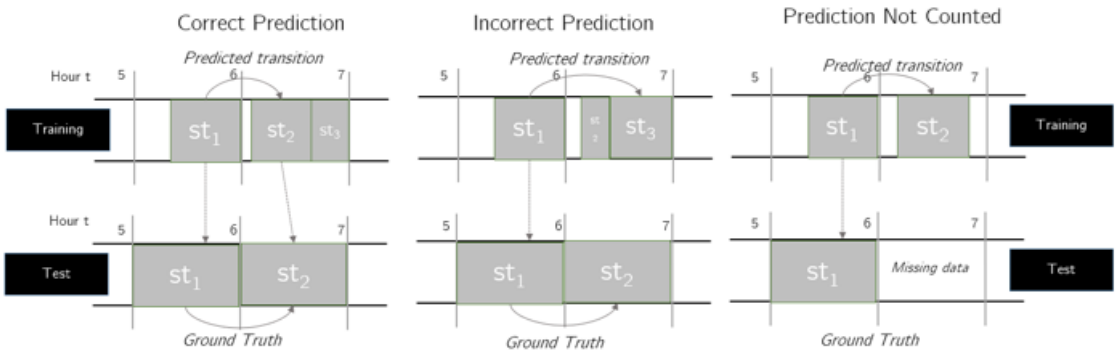

Figure 7.14: Correct(left), Incorrect(middle) or none(right) prediction scenario

The prediction is counted to be made only from the known locations. The known locations are the ones which are present in the markov model. A prediction is counted as correct, if the predicted vector with highest weight is also the one visited (present in ground truth vector) in the next hour slot, else an incorrect prediction. Consider an example as shown in figure 7.14. On the left of the 7.14, from the training model, it is suggested to transition from state st_1 to st_2 and st_3 from hour 5 to hour 6. The actual transition is made from st_1 to st_2. In this case the prediction is considered as correct as the prediction suggested to be at state st_2 with highest weight. In the middle of the 7.14, the prediction is made from st_1 to st_3 from hour 5 to hour 6 and the actual transition is made from st_1 to st_2 from hour 5 to hour 6. Hence, this is considered as the incorrect prediction. On the right, the prediction is neither correct or incorrect as there is no stay-point found in the hour 6. The predictions are rated only for the time-slots where an actual transition has been occurred.

The accuracy precision A_p is defined as:

$$A_{\mathrm{p}} = 1 - |P - GT|/2 \tag{7.2}$$

Where P is the prediction vector, GT is the ground truth vector. From the 7.14, the left most scenario yields the prediction vector states as P = [st_2, st_3] and the ground truth vector contains states as GT = [st_2, 0]. This approach calculates 1-total distance variation. The distance variation calculates the distance between the predicted values and the ground truth and hence, evaluates the quality of the prediction. To understand this approach in detail, please refer the book [10].

The cosine similarity C is defined as:

$$C = (P.GT)/(||P||||GT||) \tag{7.3}$$

The numerator is the dot product of the prediction vector P and ground truth vector GT and the denominator is the magnitude product of the two vectors. The cosine similarity C = 1 represents the vectors are exactly same, which means the prediction is 100% matching as the actual transitions. A cosine similarity C = 0 represents orthogonality, which means that the prediction is completely wrong.

The probability of transition from the current state to all the other state in the next time-slot is fetched from the markov chain model as $p = \{p_1, p_2, \ldots, p_n\}$ where p_i indicates the probability of state i. The true stay in the next time-slot is also stored as a vector $gt = \{gt_1, gt_2, \ldots, gt_n\}$ where gt_i represents the true or actual stay of state i. These two vectors are compared to determine the cosine similarity. Hence, in contrast with the accuracy correctness A_c and accuracy preciseness A_p, the cosine similarity C considers the probability of prediction from markov chain and the true duration of the stay. It means, even if the predicted transition and ground truth are same, if the prediction suggests very less duration stay in the next hour but the ground truth is for longer duration in the next hour, the cosine similarity will be smaller. Let us consider the example shown in the 7.14 on the left. The transition from st_1 to st_2 is predicted and the ground truth is also the same. The prediction is correct but to what extend? The prediction suggests a very short duration stay at state st_2 in the next hour, whereas the actual stay at hour 6 at state st_2 is longer. This will result in smaller cosine similarity value. Hence, the cosine similarity C will be close to 1 only if the duration of stay in ground truth matches with the prediction, otherwise it will be close to 0. Therefore, the cosine similarity close to 1 suggests precise predictions and the cosine similarity close to 0 suggests bad predictions.

7.2.4.2 Results

The first evaluation done is to find out the accuracy correctness and accuracy precision. For a total 182 users, the box plots are shown in figure 7.15. The mean accuracy correctness A_c for 182 users is found to be 92.7%. The mean accuracy precision A_p for 182 users is found to be 63.7%. The lower accuracy precision indicates that all the correct predictions did not predict the exact duration of the stay as the ground truth.

Another evaluation is done using cosine similarity. The cosine similarity C mean for 182 users is found to be 0.725 as shown in the figure 7.15.

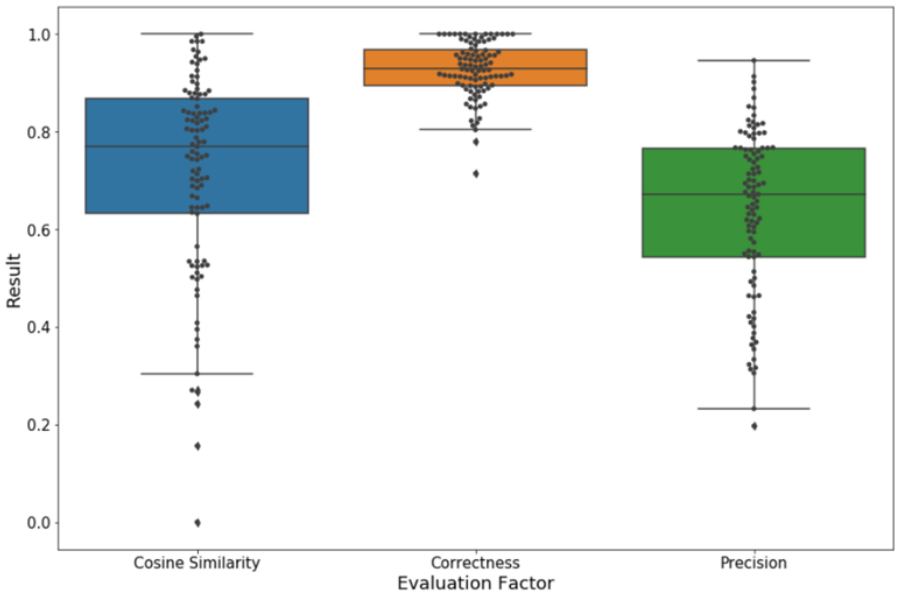

Figure 7.15: Evaluation for Geolife dataset

States	23	33	48	49	59
Time(Seconds)	9.082	16.918	26.895	28.192	28.641

Figure 7.16: Number of states vs Time(sec) on Android Application

7.2.5 PRE Application Performance

The PRE application creates markov chain based on time-slotted data. The time taken to create a markov chain model is evaluated. Since, the prototype application uses Geolife dataset, the evaluation is done with real-life dataset.

The application is run on an emulator Nexus 5X API 28 on Android Studio. The resolution of the device was 1080x1920: 420dpi, CPU x86 with 1536 MB RAM. As the number of states vary, so does the time to calculate the markov chain. The figure 7.16 lists the states versus time in seconds. The states data is taken from the users from Geolife dataset. The time for 23 states is relatively low and increases as the states increases. This evaluation is done only to indicate that state count will affect the performance of the android application. This time can be affected by many factors, for instance, emulator behaviors, cache usage and many more.

8 Conclusion and Future Work

In this chapter we summarize the thesis contribution and work, the overall evaluation results and possible aspects of future work.

8.1 Conclusion

The Location-Based Applications (LBAs) keep collecting the user location-based data. Based on the collected location information, user's whereabouts can be easily guessed. To help user understand the consequences of sharing location details with LBAs, a location prediction model is built. The markov chain model is suggested for location prediction on mobile devices.

The prediction model is used for privacy-risk estimation. An Android application called Privacy-Risk-Estimator (PRE) is developed. The PRE application predicts the future locations based on current location and location history. The locations are predicted with a confidence percentage indicating the predictability of the users. The users can use the PRE application before sharing their location information with the LBAs. A calibration for the privacy-risk estimation algorithm is also suggested.

The approach is tested and evaluated on real-life Geolife dataset from Microsoft [17] [15] [16]. The dataset contains 182 user GPS trajectory data over a period of 5 years. The algorithm resulted in 92.7% mean accuracy correctness, 63.7% mean accuracy preciseness and average cosine similarity of 0.725 for 182 different users. The evaluation result suggests that the approach can predict movements with an above average accuracy percentage.

8.2 Discussion

In this thesis, we presented an algorithm for location prediction using markov chain model. The prediction approach is simple enough to be able to run on smartphones and applicable for most users.

The thesis work had several challenges as it dealt with real-life data. It was difficult to generate a stay-point clustering algorithm as many users have missing data. The basic clustering algorithm did not result into any particular pattern. It was an important investigation that the most GPS trajectories are tracking mostly outdoor movements. As a result of this investigation, the start and end points of trajectories are considered for stay-point detection. Since, the users were real, user evaluation was an important section.

8.3 Future Work

The prediction model may have better prediction accuracy with more spatio-temporal features. For instance, building separate transition matrices for weekdays and weekends may increase the prediction accuracy for the users who have very distinctive movement patterns on weekdays and weekends. The algorithm can also be tested in future to accept more data like calendar entries or call/SMS log. As suggested by the paper [5], it can reveal important information about user's next movement. Even though, adding new features will increase the complexity of the algorithm, it could be interesting to see the effect on the prediction model accuracy, cosine similarity and privacy-risk estimation.

The user interface of PRE application can have several potential improvements. The markov chain and privacy-risk estimation algorithms are implemented on android device. But, the PRE application uses the smoothened data. This can be extended in future to accept the raw GPS points as input on the android device and create and update the markov model regularly. Currently, the time-slotted data is used as the input for the android application. The algorithms explained in previous chapters can be used as an example to extend the android application implementation. The finished android application can be published on Google Playstore to collect user feedback.

The survey for calibration of the privacy-risk estimation had only 10 participants. The survey must be conducted with more participants in order to learn the human expectations on large scale. It will be interesting to see if the there exists a general memory-loss factor for most users or it varies a lot.

Bibliography

[1] "Americans' complicated feelings about social media in an era of privacy concerns." In: *P. R. Center* (2018) (cit. on p. 14).

[2] M. Baratchi, N. Meratnia, P. J. M. Havinga, A. K. Skidmore, B. A. K. G. Toxopeus. "A Hierarchical Hidden semi-Markov Model for Modeling Mobility Data." In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. Seattle, Washington: ACM, 2014, pp. 401–412. ISBN: 978-1-4503-2968-2. DOI: 10.1145/2632048.2636068. URL: http://doi.acm.org/10.1145/2632048.2636068 (cit. on p. 22).

[3] M. Baratchi, N. Meratnia, P. Havinga, A. Skidmore, A. Toxopeus. "Sensing solutions for collecting spatio-temporal data for wildlife monitoring applications: a review." English. In: *Sensors (Switserland)* 13.5 (May 2013). eemcs-eprint-23374, pp. 6054–6088. ISSN: 1424-8220. DOI: 10.3390/s130506054 (cit. on p. 22).

[4] P. Baumann, W. Kleiminger, S. Santini. "The Influence of Temporal and Spatial Features on the Performance of Next-place Prediction Algorithms." In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '13. Zurich, Switzerland: ACM, 2013, pp. 449–458. ISBN: 978-1-4503-1770-2. DOI: 10.1145/2493432.2493467. URL: http://doi.acm.org/10.1145/2493432.2493467 (cit. on pp. 22, 23).

[5] J. B. Gomes, C. Phua, S. Krishnaswamy. "Where Will You Go? Mobile Data Mining for Next Place Prediction." In: *Proceedings of the 15th International Conference on Data Warehousing and Knowledge Discovery - Volume 8057*. DaWaK 2013. Prague, Czech Republic: Springer-Verlag New York, Inc., 2013, pp. 146–158. ISBN: 978-3-642-40130-5. DOI: 10.1007/978-3-642-40131-2_13. URL: http://dx.doi.org/10.1007/978-3-642-40131-2_13 (cit. on pp. 21, 22, 74).

[6] A. Görlach, A. Heinemann, W. W. Terpstra. "Survey on Location Privacy in Pervasive Computing." In: *Privacy, Security and Trust within the Context of Pervasive Computing*. Ed. by P. Robinson, H. Vogt, W. Wagealla. Boston, MA: Springer US, 2005, pp. 23–34. ISBN: 978-0-387-23462-5. DOI: 10.1007/0-387-23462-4_3. URL: https://doi.org/10.1007/0-387-23462-4_3 (cit. on p. 17).

[7] J. H. Kang, W. Welbourne, B. Stewart, G. Borriello. "Extracting Places from Traces of Locations." In: *Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. WMASH '04. Philadelphia, PA, USA: ACM, 2004, pp. 110–118. ISBN: 1-58113-877-6. DOI: 10.1145/1024733.1024748. URL: http://doi.acm.org/10.1145/1024733.1024748 (cit. on pp. 18, 19, 44).

[8] C. Katsaounis. *Habitat Use of the Endangered and Endemic Cretan Capricorn and Impact of Domestic Goats*. University of Twente Faculty of Geo-Information and Earth Observation (ITC), 2012. URL: https://books.google.de/books?id=YH65AQAACAAJ (cit. on p. 22).

[9] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen. "The Mobile Data Challenge: Big Data for Mobile Computing Research." In: (2012) (cit. on pp. 21, 22).

[10] D. A. Levin, Y. Peres, E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009 (cit. on p. 69).

[11] A. Noulas, S. Scellato, N. Lathia, C. Mascolo. "Mining User Mobility Features for Next Place Prediction in Location-Based Services." In: *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*. ICDM '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1038–1043. ISBN: 978-0-7695-4905-7. DOI: 10.1109/ICDM.2012.113. URL: http://dx.doi.org/10.1109/ICDM.2012.113 (cit. on pp. 20, 21).

[12] Z. Riaz, F. Dürr, K. Rothermel. "Location Privacy and Utility in Geo-social Networks: Survey and Research Challenges." In: *16th Annual Conference on Privacy, Security and Trust (PST)*. Aug. 2018, pp. 1–10. DOI: 10.1109/PST.2018.8514193 (cit. on p. 18).

[13] Z. Riaz, F. Dürr, K. Rothermel. "Understanding Vulnerabilities of Location Privacy Mechanisms Against Mobility Prediction Attacks." In: *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. MobiQuitous 2017. Melbourne, VIC, Australia: ACM, 2017, pp. 252–261. ISBN: 978-1-4503-5368-7. DOI: 10.1145/3144457.3144505. URL: http://doi.acm.org/10.1145/3144457.3144505 (cit. on pp. 17, 34).

[14] C. V. "Challenging Warrantless Gps Tracking Of State Employee's Personal Car." In: *New York State Department Of Labor* (2013) (cit. on p. 14).

[15] Y. Zheng, Q. Li, Y. Chen, X. Xie, W.-Y. Ma. "Understanding Mobility Based on GPS Data." In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp '08. Seoul, Korea: ACM, 2008, pp. 312–321. ISBN: 978-1-60558-136-1. DOI: 10.1145/1409635.1409677. URL: http://doi.acm.org/10.1145/1409635.1409677 (cit. on pp. 3, 20, 22, 57, 73).

[16] Y. Zheng, X. Xie, W.-Y. Ma. "GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory." In: *IEEE Data Eng. Bull.* 33 (June 2010), pp. 32–39 (cit. on pp. 3, 20, 22, 57, 73).

[17] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma. "Mining Interesting Locations and Travel Sequences from GPS Trajectories." In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 791–800. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526816. URL: http://doi.acm.org/10.1145/1526709.1526816 (cit. on pp. 3, 19, 20, 22, 57, 73).

All links were last followed on November 26, 2018.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature