**Design and Development of Software Agents for Location**

**Privacy-risk estimation**

Shashank Sharma

August 2018

**Abstract**

The usage of mobile devices has become ubiquitous in today’s world. The human location is shared with different application on a mobile device. This location data can be used by different third-party applications to predict user’s future locations. This is a privacy attack which makes use of user’s past location data.

In this thesis we present an algorithm which predicts user future movements with confidence percentages. This algorithm is first implemented on python using Microsoft Geolife data. This data contains 182 user trajectories data for 5 years. The same algorithm is then implemented on Android device.

The raw trajectories are used from Microsoft Geolife data. This data is first used to find the stay points. The stay points are the places where user spent at least 20 minutes within the radius of 200 m. Once the stay points are found, we form states from these stay points using snapping algorithm. Once this process is done, we start creating Markov chains. This Markov chain is then used to predict user future locations.

**Acknowledgement**

Contents

[1 Introduction 1](#_Toc521925956)

[2 Related Work 2](#_Toc521925957)

[3 System Model 2](#_Toc521925958)

[3.1 Components 2](#_Toc521925959)

[3.2 Assumptions 2](#_Toc521925960)

[3.3 Problem Statement 2](#_Toc521925961)

[4 Algorithms 2](#_Toc521925962)

[4.1 Prediction Approach (Markov Model) 2](#_Toc521925963)

[4.2 Accuracy 2](#_Toc521925964)

[4.3 Discussion and Summary 2](#_Toc521925965)

[5 Implementation 2](#_Toc521925966)

[5.1 Stay-points 2](#_Toc521925967)

[5.2 States from Stay-points 2](#_Toc521925968)

[5.3 Hourly weights of states 2](#_Toc521925969)

[5.4 Forming Markov chain on states 2](#_Toc521925970)

[5.5 Android implementation 2](#_Toc521925971)

[6 Evaluation 2](#_Toc521925972)

[6.1 User data analysis 2](#_Toc521925973)

[6.2 Discussion and Summary 2](#_Toc521925974)

[7 Conclusion and Future Work 2](#_Toc521925975)

[7.1 Summary 2](#_Toc521925976)

[7.2 Future Work 2](#_Toc521925977)

[2 Definitions 2](#_Toc521925978)

# **1 Introduction**

This algorithm is designed to predict human locations in a real-world scenario. The GPS data is taken as input and the processed using the below algorithm.

The Algorithm has several steps:

* \_ Detect stay-points (also detect start or end of the trajectory)
* \_ Group stay-points to form states
* \_ Calculate hourly weights for the states
* \_ Apply Markov chain for the data available

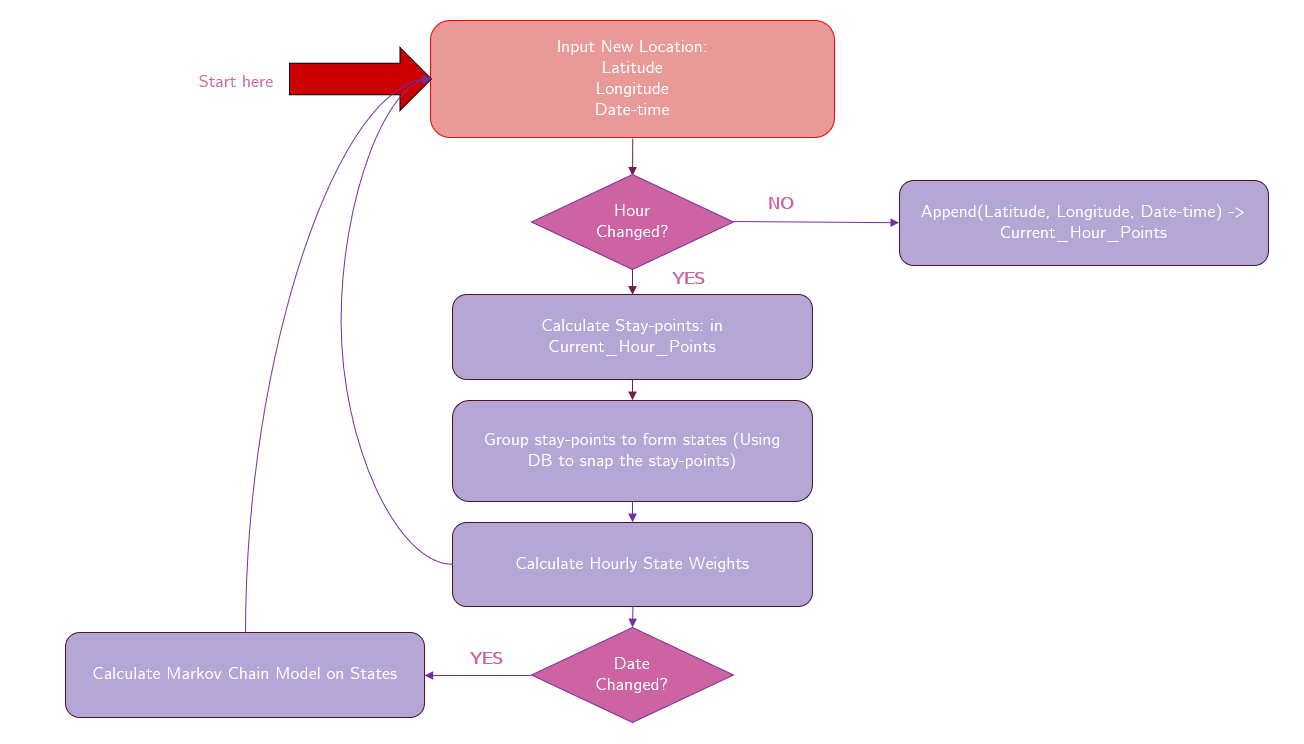


Figure 1: Algorithm Flow-chart

# 2 Related Work

# 3 System Model

## 3.1 Components

## 3.2 Assumptions

## 3.3 Problem Statement

# 4 Algorithms

## 4.1 Prediction Approach (Markov Model)

## 4.2 Accuracy

## 4.3 Discussion and Summary

# 5 Implementation

## 5.1 Stay-points

## 5.2 States from Stay-points

## 5.3 Hourly weights of states

## 5.4 Forming Markov chain on states

## 5.5 Android implementation

# 6 Evaluation

## 6.1 User data analysis

## 6.2 Discussion and Summary

# 7 Conclusion and Future Work

## 7.1 Summary

## 7.2 Future Work

# 2 Definitions

\_ Stay-points: Stay-points are any points which are stayed by the user in

user trajectories or it is the start or the end of the trajectory. For example,

if user start at his home, the home itself is a stay-point. Now he moves

towards work, but he visits a cafe in between for breakfast. The cafe is

also, a stay-point and then he finishes his trajectory at work, where work

is again a stay-point. The places like cafe in this case is identified using

distance and time-based clustering. For example, a set of points within

200m with total duration of stay greater than 20 minutes can be regarded

as a stay-point within the trajectory.

\_ State: A state is formed using a group of stay-points. This is done using

a distance threshold for states. All the stay-points within this threshold

distance are grouped together as a single state. This is called snapping

stay-points to the states. The mean of all location latitudes and longitudes

from stay-points within a state are stored per state. Finally Markov Chain

model is applied to the states. Note: A new stay-point is only added to the

state if after calculating the mean of the new state, all the existing stay-

points still stay within the distance threshold from this mean. This is done

to avoid drifting problem while aggregating the stay-points into states.

Algorithm 1 Read new location and process

1: Read new latitude, new longitude, new datetime information and process

these new points

2: while NewLocationDetected == True do

3: Set newHour = datetime:hour

4: Set newDate = datetime:date

5: if newHour! = prevHour then

6: prevHour newHour

7: calculateLastHourStayPoints()

8: formStates()

9: calculateStateLastHourWeights()

10: end if

11: if newDate! = prevDate then

12: prevDate newDate

13: recalculateMarkovModel()

14: end if

15: end while

2

Algorithm 2 calculateLastHourStayPoints() : Calculate last hour stay-points

1: Calculate stay-points

2: trackingThreshold: Maximum time distance between two points

3: thresholdDistance: Stay-point threshold distance covered

4: thresholdTime: Stay-point threshold time spent

5: for eachLastHourPoint do

6: if (point(i):datetime 􀀀 point(i 􀀀 1):datetime) >= trackingThreshold

then

7: Add point(i), point(i-1) as stay-points

8: recalculateStartEndStaypoint()

9: end if

10: if distance(point(i); cluster) <= thresholdDistance then

11: add point i to cluster

12: calculate Cluster Means

13: else

14: if (cluster! = empty) And duration(cluster) >= thresholdTime

then

15: Add this cluster to stay-points

16: recalculateStartEndStaypoint()

17: end if

18: end if

19: end for

Algorithm 3 recalculateStartEndStaypoint() : Calculte start-end of staypoints

1: for eachStaypoint do

2: Set distance distance(staypoint(i); staypoint(i + 1)

3: Set time timeDifference(staypoint(i); staypoint(i + 1)

4: Set AvgSpeed distance=time

5: Set AddTime min(distance, thresholdDistance)=AvgSpeed

6: Set endT imeofStaypoint(i) endTimeStaypoint(i) + AddTime

7: Set startT imeofStaypoint(i + 1) startTimeStaypoin(i + 1) +

AddTime

8: end for

3

Algorithm 4 formStates() : Form states from stay-points

1: Form each unique stay-point as individual state

2: Now, within this loop, start combining the states

3: thresholdState: State distance threshold

4: for eachState do

5: if distance(state(i); state(i + 1)) <= thresholdDistance then

6: calculate new state mean latitude, mean longitude

7: if distance(allExitingState(i)Staypoints;NewMeanLatLong) <=

thresholdState then

8: combine state i, i+1

9: calculate State Means

10: end if

11: end if

12: end for

Algorithm 5 calculateStateLastHourWeights() : Calculate Hourly Weights of

Statesudl

1: This creates a weights of all states from 0 Hrs to 24 Hrs for each date

2: for eachState do

3: if (HourChanges) Or (StateIDChanges then

4: Calculate the start and end of the state i

5: end if

6: end for

Algorithm 6 recalculateMarkovModel() : Recalculate the Markov Model

1: This algorithm creates the transition probabilities from state i to i+1 from

hour h to h+1

2: for each Hth 􀀀 hour from 0 􀀀 24 do

3: for each ith 􀀀 state in state 􀀀 hourly 􀀀 weight do

4: state(i)􀀀 > state(i + 1) transition for H 􀀀 hour = Matrix[state] \*

Matrix[State + 1]

5: end for

6: end for

4