User location estimation in app based on location service

Pan Zicheng(predict and improvement) Zhang Yuheng(simulation and data processing)

1. Introduction

In recent years,with the ubiquitous adoption of smartphones and mobile devices,it is now common practice for one's location to be sensed,collected and likely shared through social platforms.

Their location information may lead to information leaks about the whereabouts of other users,through the co-location of events when two users are at the same location at the same time and other side information.

Here,our main work is to use the information to predict the location of user at a time.

Figure 1.

2. Data Processing

First,we define the GPS event and Meeting event:

GPS event:define as (i,t,p),agent i is in location p when time is t.

Meeting event:define as (i,j,t),agent i and j are in same location when time is t.

Then we randomly generate several agents,every agent will randomly walk in random time with a speed,the speed can be any value,for convenience,we assume it is unit,that is unit length per unit time.

So we can get the track of every agent,but the system never get it,it will be used to evaluation.From the track we can get some GPS events,which will upload to the system.

After we get the GPS events,we can use them to find the Meeting events,for example,we can get:

Then we need to map the events to nodes and edges in graph,for example:

The blue points show GPS events,the red points show Meeting events.We also define the edges as follows:

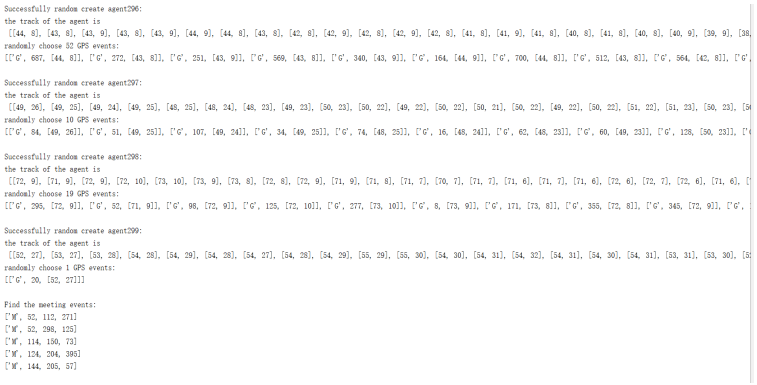

Figure 2.

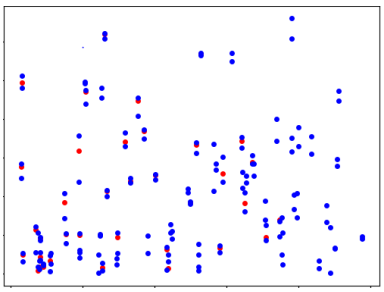

Figure 3.

Two events χ,$\chi^{'}$, with associated times t and $t^{'}$,define an edge (χ,$\chi^{'}$),if they involve a common agent i,and agent i was not involved in any events (meetings or GPS events) at times between t and $t^{'}$.

We also assign the weight $w(\chi,\chi^{'}) = v_i|t - t^{'}|$ to the edge χ,$\chi^{'}$, v_i is the maximum speed of agent i.

3. Predict and Evaluation

3.1. based algorithm

Now we already have a graph with nodes and weighted edges, but how can we use it to predict the location of events?

Obviously, if a GPS event connect to a Meeting event with an edge weighted n, because the agent can mostly move

to a place distancced n, it certainly should be located in a circle with radius n from the original GPS event.

According to what we have explained, we can obtain an algorithm to calculate the boudary of each event using Dijkstra shortest path algorithm.

3.2. the difficulty with Euclidean distance

To begin with, in real world, a map has two dimensions. So it certainly reminds us of L_2 norm, it is accurate and easy to build model. However, if solution not exists, the complexity becomes NP-Hard, which can be reduced from NP-Complete problem unit-length linkage realizability.

3.3. boundary

So we use a clever method, we calculate the distance in the x direction and y direction respectively, and treat the Euclidean distance as a one-dimensional distance. In this way, although we make the algorithm less accurate, we can avoid the case of excessive complexity.

Algorithm 1 shows the steps to predict the boundary of a specific event. Because we can get the accurate location of a GPS event, so of course we can connect 0 and each GPS event and initialize the shortest distance of it as its coordinate. It is clearly that these distance will never update when using $Dijkstra$ Algorithm.

Algorithm 1 Prediction

```
for i in GPS events do
    Connent 0 and i
    Initialize set detected+={i}
    Initialize set distance[i]=i.location
    Dijkstra Algorithm
end for
right_bound=distance
```

After calculation, now we get the set of right bound of each GPS event. Do the same steps four times and we can get the predict range.

3.4. evaluation

Since we obtain the predict range of all the agents, we now consider how to evaluate our result.

An common method is $hit\ rate$, if an event is located in the range we predict, we call it hit, or it means $miss$. However, by this algorithm we actually expand the real range. Projecting Euclidean distance to one-dimension expands the area from a circle to a foursquare, which means the $hit\ rate$ will always be 100 percent.

Unfortunately, this standard can not satisfy us. If the predict range of an event is expanding to the whole map, the $hit\ rate$ is still 100 percent, but it is meaningless.

In order to better evaluate the result, we define $accuracy = \frac{hit\ rate}{predict\ range}$, it can show how accurate the result is. We define $predict\ range = \sqrt{predict\ aera}$, so the standard can show the accurate of our algorithm and if there exists an inaccurate predict range, it will not lead to a low point for whole process.

4. Defection and Improvements

4.1. defection

In the algorithm provided, in fact, there exists a big defection. If the events path we predict has a long path of Meeting events, the predict area will expand larger and larger. If there exist one event whose time is far from the one near it, it may lead that the events in whole link use the whole map as their predict aera. On this condition, the result of the algorithm is depressed.

What's more? Consider the walking logic of agents, rarely an agent will walk still without change direction. So most time the agent can't walk to the boundary we predict.

So to conclude, the algorithm in the model is over ranged, it will provide absolute perfect $hit\ rate$, but it may lead to meaningless data.

4.2. imptovement with alpha attenuation

Then we think about how to make some improvement to over come the problem above. We consider the probability about the position that agents show up, it is clearly that the probability is much lower when approaching the boundary.

So we come out a new algorithm named alpha attenuation, we count the depth about Meeting event in each link. Then when calculating, we multiply α^n with the weight of the edge. In this way, we sacrify $hit\ rate$ a little but improve $accuracy$ a lot.

5. Conclusion and Future Work

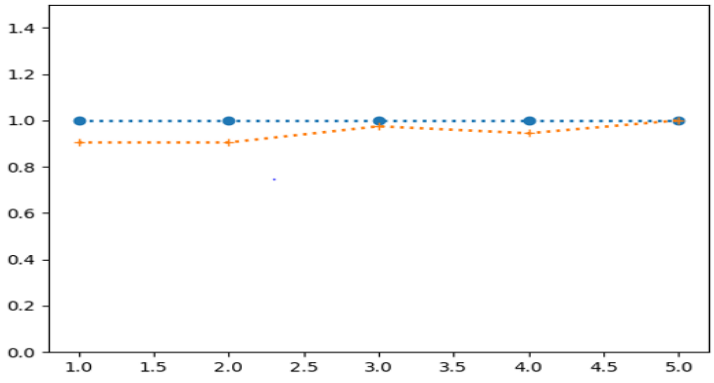

Figure 4.

We realize the algorithm given in [1] and make our own improvement. As the results show, the $hit\ rate$ is a little lower, but it is still larger than 0.9. However, $accuracy$ is much improved. So we think the algorithm we design is effective because we consider accurate data more valuable.

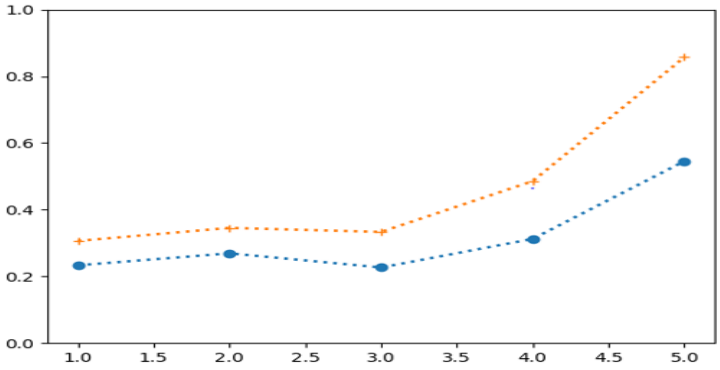

Figure 5.

For future work, first we want to design a method to help avoid leaking position under protocols. Most protection about position try to use uncertain random variable to misunderstand the system, but we think it is not what we want. What's more, we'd like to try machine learning when simulate agents so that it may be more convictive.

References

[1] M. Schaefer. Realizability of graph sand linkages. In J. Pach, editor, Thirty Essays on Geometric Graph Theory,pages461–482.SpringerNewYork,NewYork,NY, 2013

[2] Are Friends of My Friends Too Social? Limitations of Location Privacy in a Socially-Connected World Boris Aronov , Alon Efrat , Ming Li , Jie Gao , Joseph S. B. Mitchell , Valentin Polishchuk , Boyang Wang , Hanyu , Jiaxin Ding

[3] G. Argyros, T. Petsios, S. Sivakorn, A. Keromytis, and J. Polakis. Evaluating thePrivacyGuaranteesofLocationProximityServices. ACM Transactions on Privacy and Security,2017.

[4] H. Wu and Y.-C. Hu. Location Privacy with Randomness Consistency. In Proceedings on Privacy Enhancing Technologies,2016.

[5] R.Shokri,G.Theodorakopoulos,G.Danezis,J.-P.Hubaux,andJ.-Y.L.Boudec. QuantifyingLocationPrivacy: TheCaseofSporadicLocationExposure. InProc. of Privacy Enhancing Technologies (PETS),2011.