UNSW
THE UNIVERSITY OF NEW SOUTH WALES

COMP9900 - IT Project
Term 1 - 2024

Final Report

Trajectory Recovery from Ash: In the Age of AI

Jiamei Yang

Chao Xu

Chin Pok Leung

Tsz Lik Wong

Qiuzi Chu

Zhou Fang

Submission Date: 19 / 04 / 2024

Content

1 Overview

1.1 Project Background

In the 21st century, data breaches have become increasingly prevalent as advanced telecommunication technologies continue to permeate everyday human life. Mobility data, which encapsulates the daily social activities and online engagements of individuals, is routinely collected through cellular networks and various applications. This data, while integral to modern conveniences, is susceptible to unauthorized access and misuse.

A report by UpGuard revealed a significant data breach at OPTUS, one of Australia's largest telecommunication companies, in September 2022. This incident compromised the personal information of 2.2 million customers, including names, addresses, and government ID numbers. Further compounding the issue, data from The Office of the Australian Information Commissioner (OAIC) indicates a 19% increase in cybersecurity incidents from January

2022 to December 2023, escalating from 46 to 97 incidents. Nearly two-thirds of these were classified as “malicious and criminal attacks,” leading to potential leaks of sensitive information such as contact details and identities. Alarmingly, much of this data leakage occurs without our knowledge.

Consider a typical day for a college student: waking up in an apartment, watching local news on a mobile device, connecting to campus Wi-Fi, and ordering dinner from a local diner via a self-ordering kiosk. These routine activities, while seemingly benign, pose significant risks to personal privacy and data security. Each interaction, from connecting to public Wi-Fi to enabling location services on an app, contributes to a digital footprint. This footprint can be tracked, analyzed, and, in the wrong hands, exploited.

Despite awareness of the threats posed by cyber-attacks, traditional data protection methods employed by companies, such as collecting location data via base stations or intentionally adding bias to positions, have proven largely ineffective.

Our research indicates that even these aggregated or "encrypted" data points can be reassembled by hackers to track individual trajectories, particularly with the aid of advanced AI-driven neural networks designed for such attacks.

Consider the daily routine of a typical college student: from using campus Wi-Fi to dining at local eateries, each activity contributes to a digital footprint that is not only extensive but consistently patterned. This consistency in daily movements, such as regular visits to a home or campus, creates a predictable pattern that can be exploited. For instance, a student may prefer dining at KFC over Hungry Jacks, but is unlikely to travel hundreds of miles for a meal. By analyzing frequently visited locations, it is possible to construct a unique and private route profile for individuals. With enough data points and minimal semantic information like "the top location is apartment A and the second is university B," an individual’s movements and personal details can be precisely mapped and retrieved.

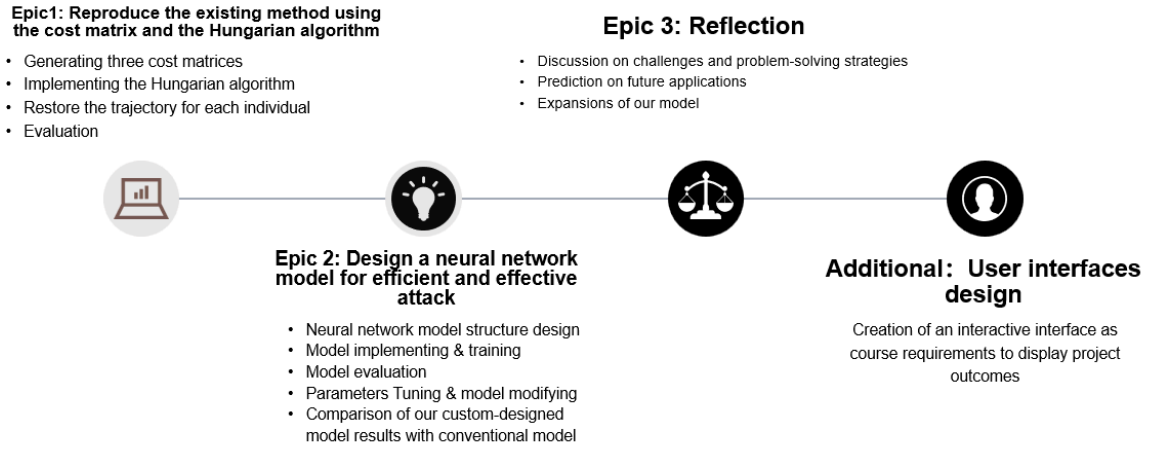
Moreover, in the age of AI, neural networks can further enhance this capability by learning to recognize physical world features like barriers, rivers, and mountains, and understanding the inherent behavior of humans or vehicles from scattered data points. This profound capability underscores the urgent need for innovative approaches to protect location data against unauthorized access and ensure personal privacy in an increasingly digital world.

1.2 Project Objective

Given the profound vulnerabilities highlighted in our background analysis concerning data privacy in the digital age, our project is strategically designed to confront these challenges head-on. As technology evolves, so too must our approaches to safeguarding personal information. In this light, our project's objectives are carefully structured to develop and evaluate advanced solutions that address these pressing issues.

Throughout our project, each functionality developed has been strategically aligned with the initial requirements provided, ensuring that our solutions are not only innovative but also directly address the specific needs outlined at the project's inception.

Our project can be broadly divided into four major sections:



Epic 1: Trajectory Analysis and Reproduction

In this segment of the project, we focused on reproducing the established method which leverages both the cost matrix and the Hungarian algorithm to reconstruct individual trajectories from aggregated data. The process began with generating three distinct cost matrices that encapsulate different dimensions and perspectives of the data. Subsequently, the Hungarian algorithm was applied to these matrices to restore the trajectory for each individual accurately. The efficacy of this reproduction was then thoroughly evaluated to ensure fidelity to the original methodology and to assess any deviations or enhancements needed.

This functionality directly corresponds to the first requirement to develop a functional implementation of the described attack. By using the cost matrix and Hungarian algorithm, we adhere closely to the description in the original study, reproducing the attack method with high fidelity.

Epic 2: Deep Learning Model for Enhanced Attack Efficacy

The second major part of our project involved the development of a deep learning model designed to surpass the efficiency and effectiveness of traditional attack methods. This involved creating a bespoke neural network architecture that was meticulously compared to conventional models such as the LSTM. Through careful design, implementation, and training phases, we tailored the model to meet the specific challenges presented by our datasets. The performance of our model was rigorously evaluated to ensure it met the high standards required for both precision and reliability in privacy protection.

Our development of a deep-learning-based alternative to the traditional attack methods aligns with the requirement to innovate beyond the existing solutions. The model was designed to be more efficient and effective, representing our main contribution to advancing the field.

We constructed a framework for comparing this novel model against the original attack, which meets the criteria of evaluating both methods across chosen datasets using the same metrics.

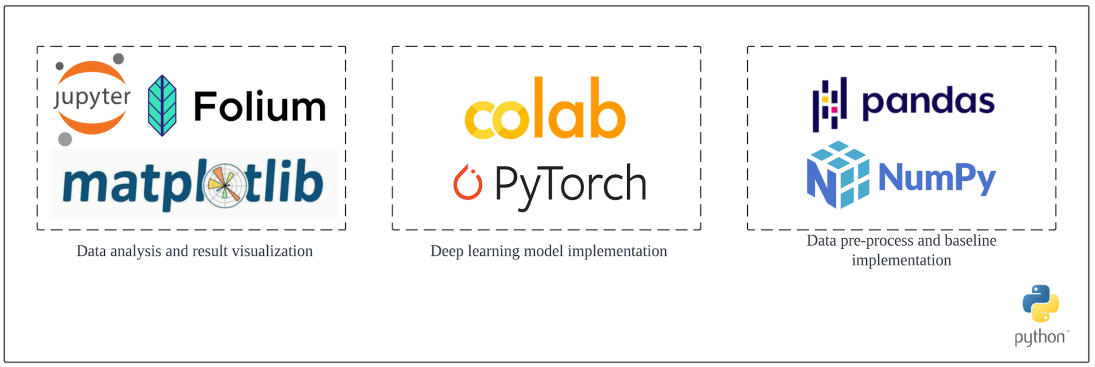
Additional: User Interface Design
Recognizing the importance of user engagement and the need to effectively communicate our findings, this part of our project was dedicated to the design of a user-friendly web interface.

This interface was crafted not only to showcase the results of our project but also to facilitate interaction with the end users. It serves as a practical demonstration of how our research can be applied in real-world scenarios, enhancing the accessibility and utility of our work.

Epic 3: Project Reflection and Future Insights
Our third epic served as a reflective culmination of the project, where we analyzed the encountered challenges and our strategic responses. This section not only discusses the technical and logistical hurdles but also ventures into the broader implications of our work. We predict future applications of our findings, considering how they might be expanded or modified in subsequent studies. This reflection is pivotal in understanding the practical impact of our work and guiding future efforts in the field.

This epic, while more reflective and analytical, supports the overarching project requirement of adapting to progress and challenges encountered during the baseline implementation. It provides insights and directions for future enhancements and research.

1.3 Overall System Architecture



Since our project is focused on the algorithm, our system does not have a layered architecture given in the lecture slides. Hence, we are presenting our proposed tech stack in this section.

For data pre-processing scripts and baseline implementation, the Pandas and NumPy library will be used.

For deep learning, we plan to implement our models using the PyTorch framework and train our models on Google Colab with GPU.

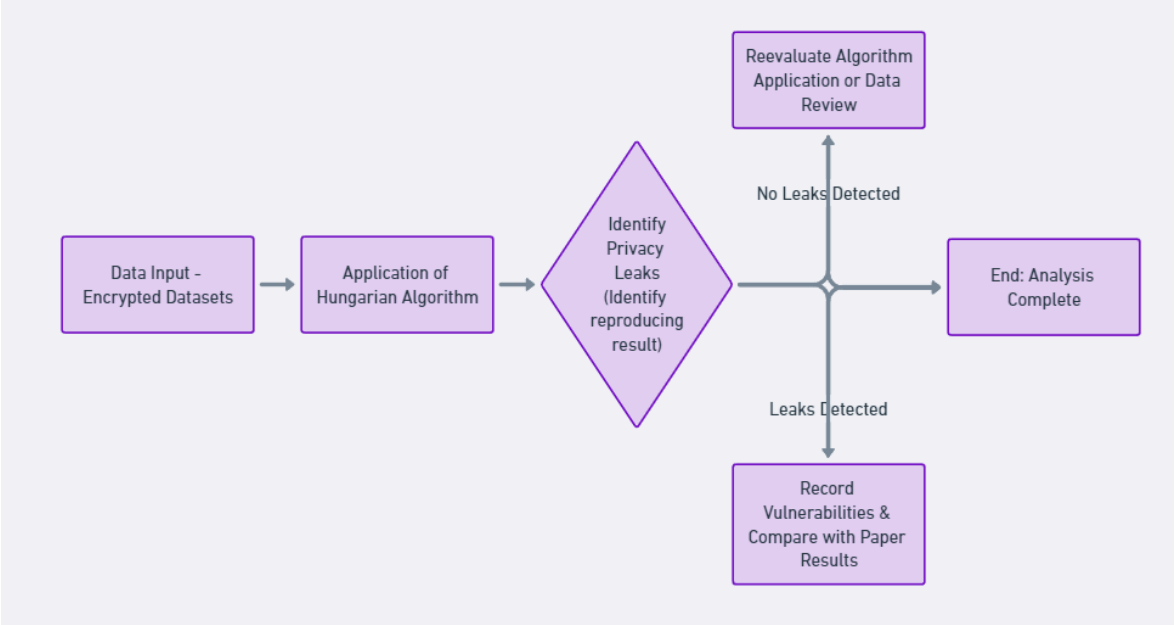
Finally, for data analysis and visualization of evaluation metrics: Matplotlib is used to generate figures; and Folium is used to visualize trajectories on a map. Relevant scripts will be implemented in Jupyter notebooks for better user experience and documentation.

2 Core Components and Deliverables

2.1 User Story and Mapping

Our project was meticulously designed to reflect the evolving needs of our clients and the broader implications of data privacy in the digital age. At the outset, we introduced a series of user stories that provided a narrative framework reflecting real-world scenarios and challenges faced by individuals in their daily interactions with digital technologies. These stories were instrumental in shaping the direction and focus of our project, ensuring that every development phase was aligned with the user's needs and client's requirements.

Reproduction of results in 'Trajectory Recovery From Ash' (Xu et al. 2017)
User Story: As a security researcher, I would like to reproduce the trajectory recovery attack using the algorithm described in 'Trajectory Recovery From Ash' (Xu et al. 2017) on at least 2 open source mobility datasets.
Acceptance Criteria: - The algorithm in 'Trajectory Recovery From Ash' (Xu et al. 2017) shall be implemented. - The algorithm shall be evaluated on at least 2 open source datasets.



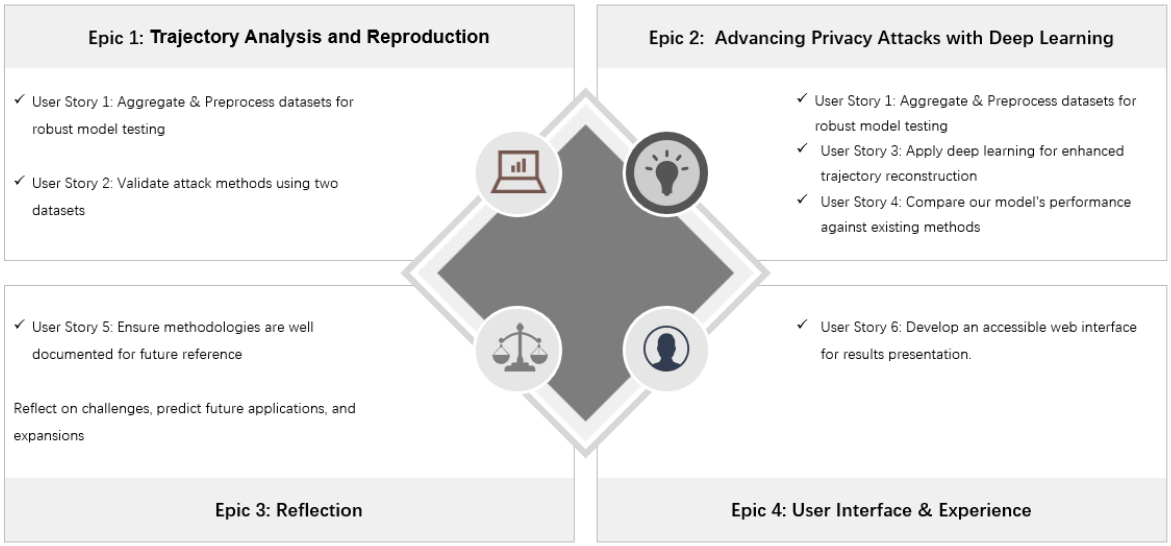
User Story:

As a security researcher, I would like to create aggregated location datasets from at least 2 open source datasets.

Acceptance Criteria:

- Scripts shall be implemented to convert at least 2 open source mobility datasets to aggregated location datasets with a standard format.
- The created aggregated location datasets should contain anonymized user identifiers as ground truth for evaluation.

By integrating user stories throughout the project lifecycle, we ensured that each development phase was not only technically sound but also deeply rooted in practical, real-world applications. This approach has not only satisfied our client's requirements but also enhanced the project's relevance and impact



2.2 Datasets

2.2.1 Shanghai Telecom Dataset

2.2.1.1 Overview

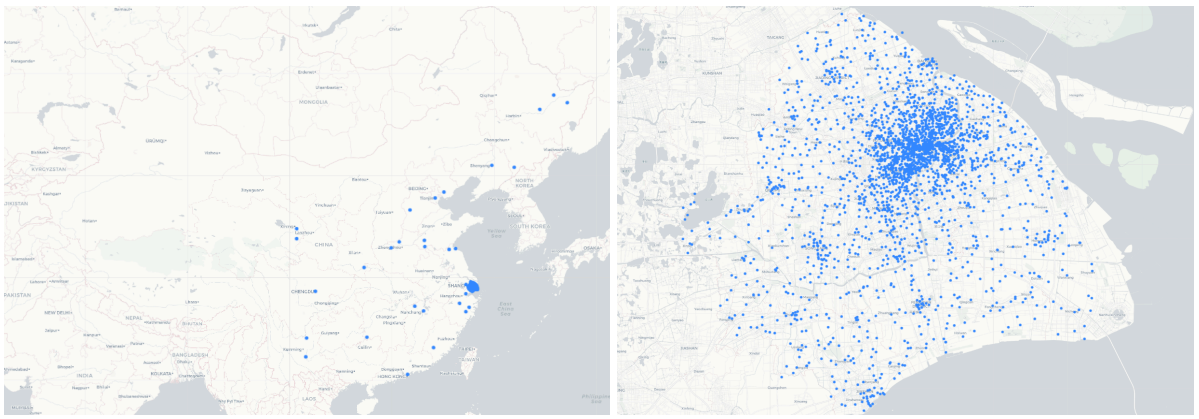
The Shanghai Telecom dataset (Li et al. 2022; Guo et al. 2020; Wang et al. 2021) records the Internet activity of mobile users using Shanghai Telecom's service in Shanghai, 2014. Each record contains a date, begin time and end time of Internet access, location of the interacted base station, and an anonymized user identifier.

Source	Mobile Internet access
Time	2014
Duration	6 months
User number	9481
Base station number	3233

Table 1. Key statistics of the Shanghai Telecom dataset

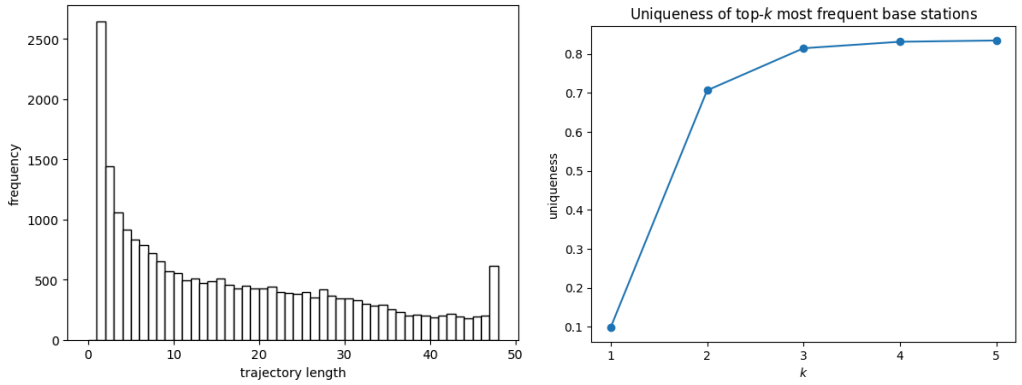
2.2.1.2 Exploratory Analysis

A subset of records from 2nd June 2014 to 9th June 2014 are selected for the below analysis.



Distribution of base stations
left: all base stations
right: basestations in Shanghai

From the left figure above, the majority of the base stations are located in Shanghai. Records outside of Shanghai are removed in subsequent analysis. The right figure above illustrates the distribution of base stations in Shanghai.



left: Length of recovered sub-trajectories (30-min temporal resolution)
right: Uniqueness of recovered trajectories (30-min temporal resolution)

Records from 2nd June 2014 to 8th June 2014 are aggregated into trajectories of individual users. From the left figure, most of the trajectories are short and not continuous. Our analysis agrees with the observation of sparse trajectories in 'Complete trajectory reconstruction from mobile phone data' (Chen et al. 2019). Furthermore, this explains why linear interpolation is needed in the data pre-process in 'Trajectory Recovery From Ash', and the motivation behind an additional step 3 in 'Trajectory Recovery From Ash' due to the discontinunity of trajectories (Xu et al. 2017).

The right figure shows that more than 80% of the users can be identified by their top-*k* most frequently visited base stations in the recovered trajectories. This coincides with section 3.2 in 'Trajectory Recovery From Ash' (Xu et al. 2017).

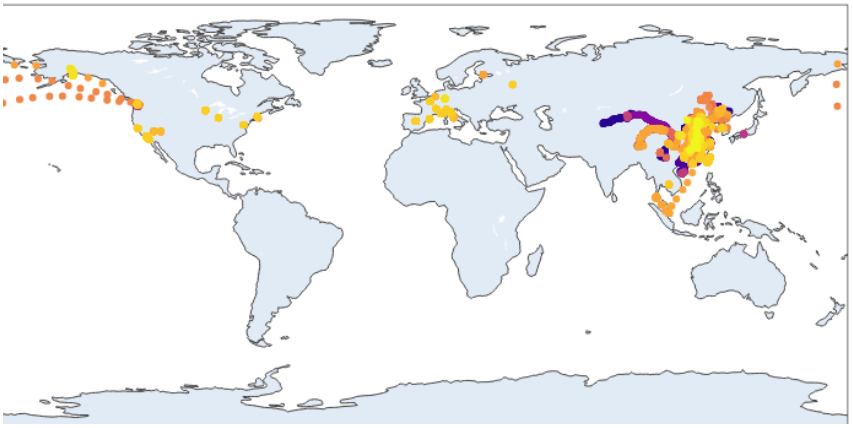
2.2.2 Geolife Dataset

The Geolife dataset (Yu et al. 2008; Yu et al. 2009; Yu et al. 2010) was collected by 182 users over 3 years with information of their GPS location. The dataset contains a variety of sampling rates. Most of the data are collected in a dense representation, e.g. every 1~5 seconds. This dataset includes life routines and also entertainment and sports activities.

Source	GPS loggers and GPS-phones
Time	Apr 2007 - Aug 2012
Duration	3 years
User number	182
Total Trajectories number	17,621
Total Distance	1.2 million kilometers
Total duration	48000+ hours

Key statistics of the Geolife dataset

By plotting the trajectory of the dataset, it shows that all records are scattered worldwidely, which may have included travels that are not regular. And as the plot shown below, the most dense area is Beijing, China. Thus, it is considerable to restrict the range for both longitude and latitude.

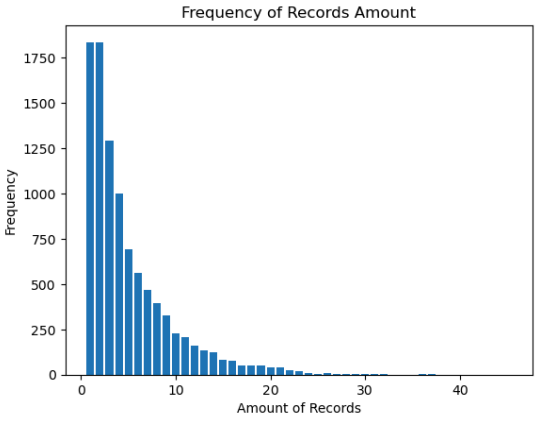


Distribution of accurate individual location points in Geolife dataset (Without a bounding box)

Here is the result after adding a bounding box for this dataset, and the only area is considered is Beijing, China. This step is considered feasible as only around 16% records are removed after this process, even though the overall boundaries are reduced heavily.

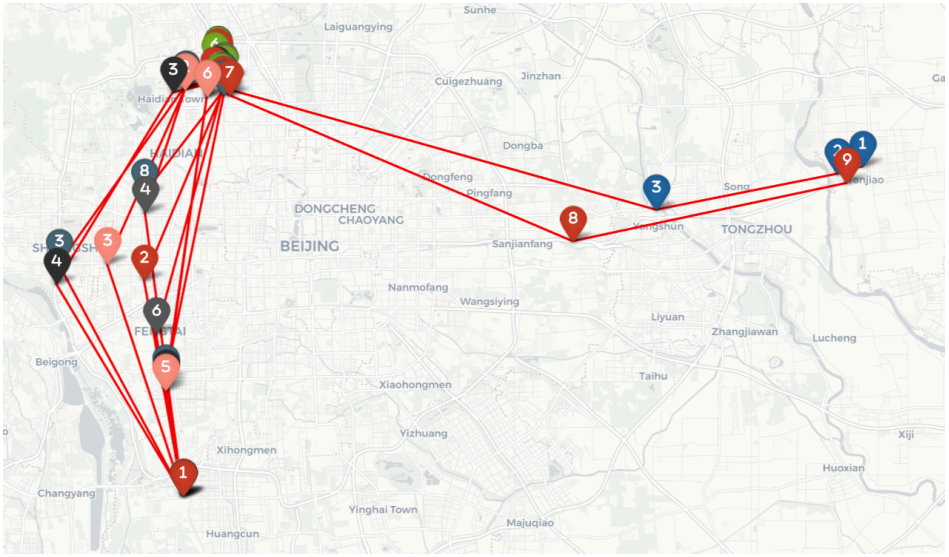
Distribution of accurate individual location points in Beijing (After adding a bounding box)

However, the amount of daily record for a user is expected to be larger than 5 in order to obtain a reasonable trajectory recovery. And as the figure shown below, a huge amount of records would not be taken into account. And by removing those data, the final size for this dataset drops to 35,500 records and 150 users.



Frequency distribution of record counts

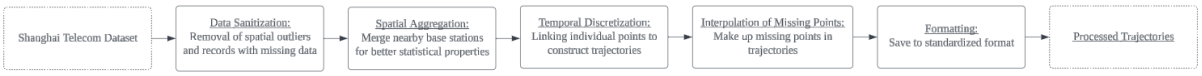
Here is the map of consecutive 7 days trajectory for a user, which reveals the regular travel of this user. The number demonstrates the order of trajectories while the color represents the days respectively. This map also proves that the daily trajectory is predictable and on the contrary, the trajectory can be recovered once the behavior pattern of the user is captured.



Consecutive 7-day trajectories of a user

2.3 Data pre-processing

2.3.1 Shanghai Telecom Dataset

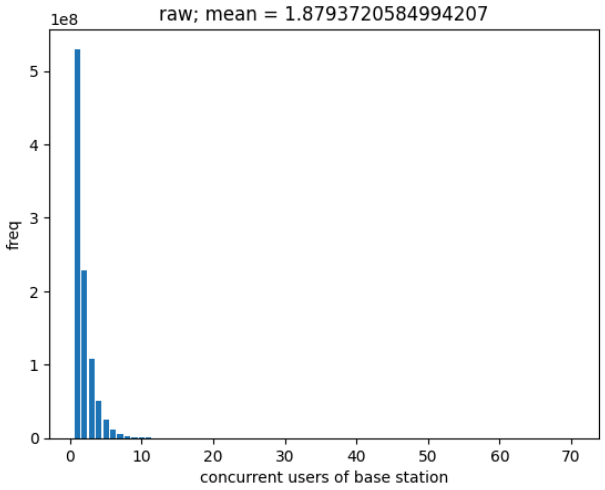


Preprocessing steps for Shanghai Dataset

2.3.1.1 Data Sanitization

In this stage, we remove all records with missing base station data. And we remove records with base station outside of the bounding box: $30.69 \leq latitude \leq 31.51$ and $120.9 \leq longitude \leq 121.9$.

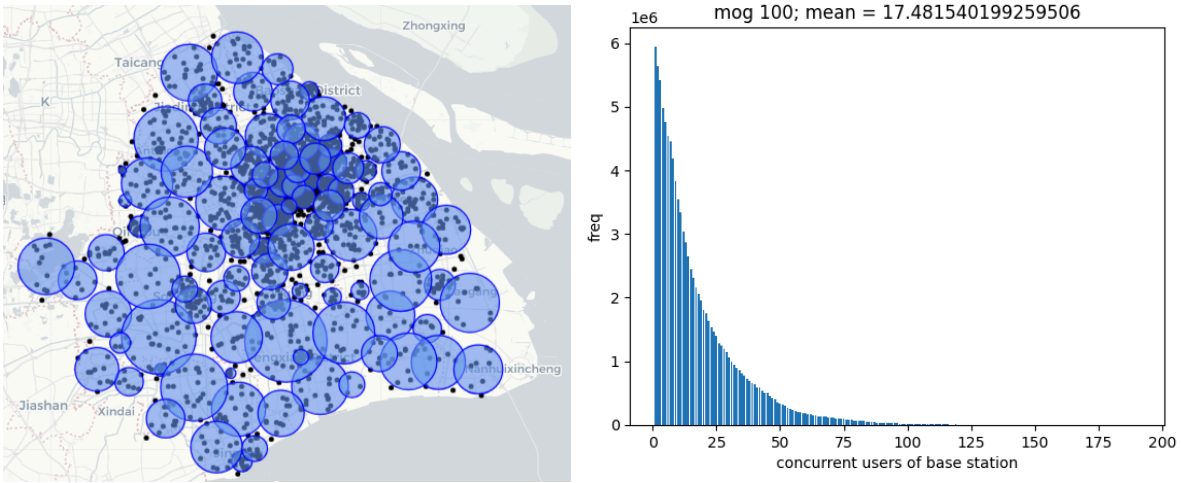
2.3.1.2 Spatial Aggregation



Distribution of the no. of concurrent users of each base station

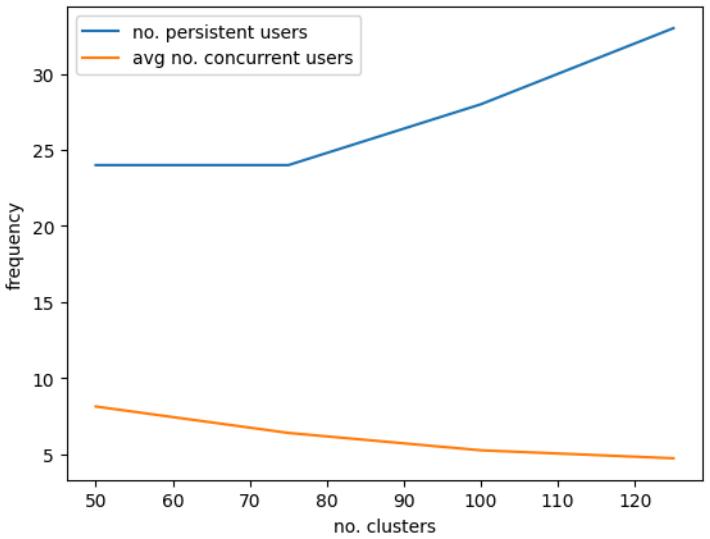
We observe that there is only a small number of users connected to the same base station at any given time, as seen in the above figure. We think that this task will be too easy for the trajectory recovery algorithm as most users can be uniquely identified by their connected base station. And thus not suitable to assess the ability of the algorithm to reconstruct complete trajectories only from aggregated datasets.

Hence, we aggregate nearby base stations to create artificial base stations with a higher number of concurrent users. To compute high quality clusters, we applied the Gaussian mixture algorithm in the scikit-learn package with spherical variance on the set of unique base stations. The Gaussian mixture algorithm consists of 2 steps: 1) compute k-means clustering to obtain the initial clusters; 2) iterative expectation maximization. The distribution of resultant clusters visually resembles the original base stations in the dataset, where the density is higher in urban areas.



Result of clustering
left: visualization of the computed clusters, radius is approximately 2 standard deviations
right: distribution of the no. of concurrent users after aggregation

With the pre-computed clusters, we use the nearest neighbor algorithm in scikit-learn to assign cluster membership to each base station. The reasons for using nearest neighbors are: 1) to avoid the ambiguity of overlapping clusters; 2) take advantage of sub-linear time computation using index pre-computed by scikit-learn.



Effects of the no. of clusters on the no. of non-stationary users and no. of concurrent users

To choose a good number of clusters, we plot the effects of the number of clusters on the number of non-stationary persistent users (users who have been to at least 3 locations every day in a chosen week), and the average number of concurrent users. We observed that using a small number of clusters would increase the probability of base stations being assigned to the same base stations, resulting in users appearing to be stationary as the base stations in their trajectories being assigned in the same cluster. Since stationary users are not desirable for assessing the day time recovery part of the algorithm, we ended up using 100 clusters which we believe is a good balance between reducing identifying power of individual base stations, and preserving movement of users across base stations.

2.3.1.3 Temporal Discretization

We follow the steps described by (Xu et al. 2017).We discretize each single day into 30-minute intervals, resulting in 48 time slots.

In each time slot, we select the location where the user stays the longest as the location of the user at that time slot.

2.3.1.4 Interpolation of Missing Points

From the plot of distribution of sub-trajectory length in Section 2.2.1.2, a majority of the trajectories contain less than 5 points per day. This sparsity cannot justify the use of linear interpolation in the paper (Xu et al. 2017). Hence, we use the last known positions of the user to predict locations at missing time slots.

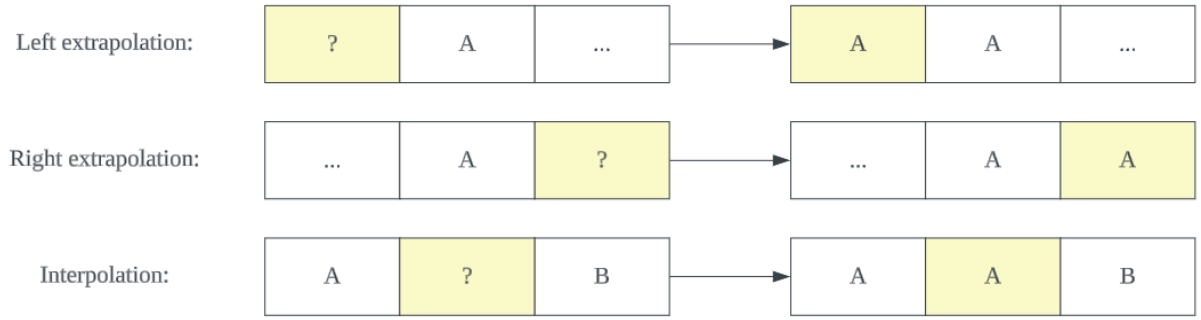


Illustration of our interpolation algorithm

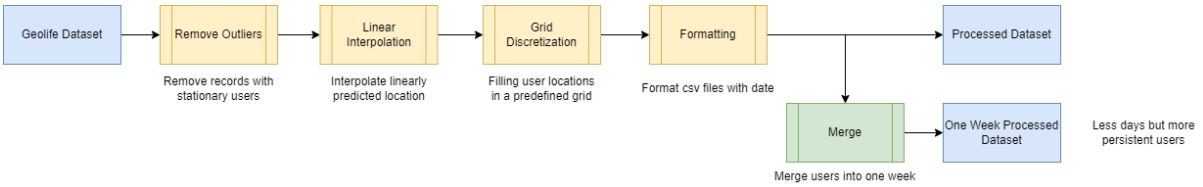
2.3.1.5 Formatting

The processed trajectories are written to standardized csv formats. The trajectories in different dates are placed in separate csv files. The columns of the csv files are described below.

Column name	Type	Description
uid	string	unique identifier of the user
t	integer	timestamp of the point
lat	float	latitude of the point
long	float	longitude of the point

2.3.2 Geolife Dataset

For this dataset, we are following a similar processing strategy. However, the issue for Geolife is its small dataset size. Although we retrieved a large number of records, we actually have a small number of users as well as trajectories.



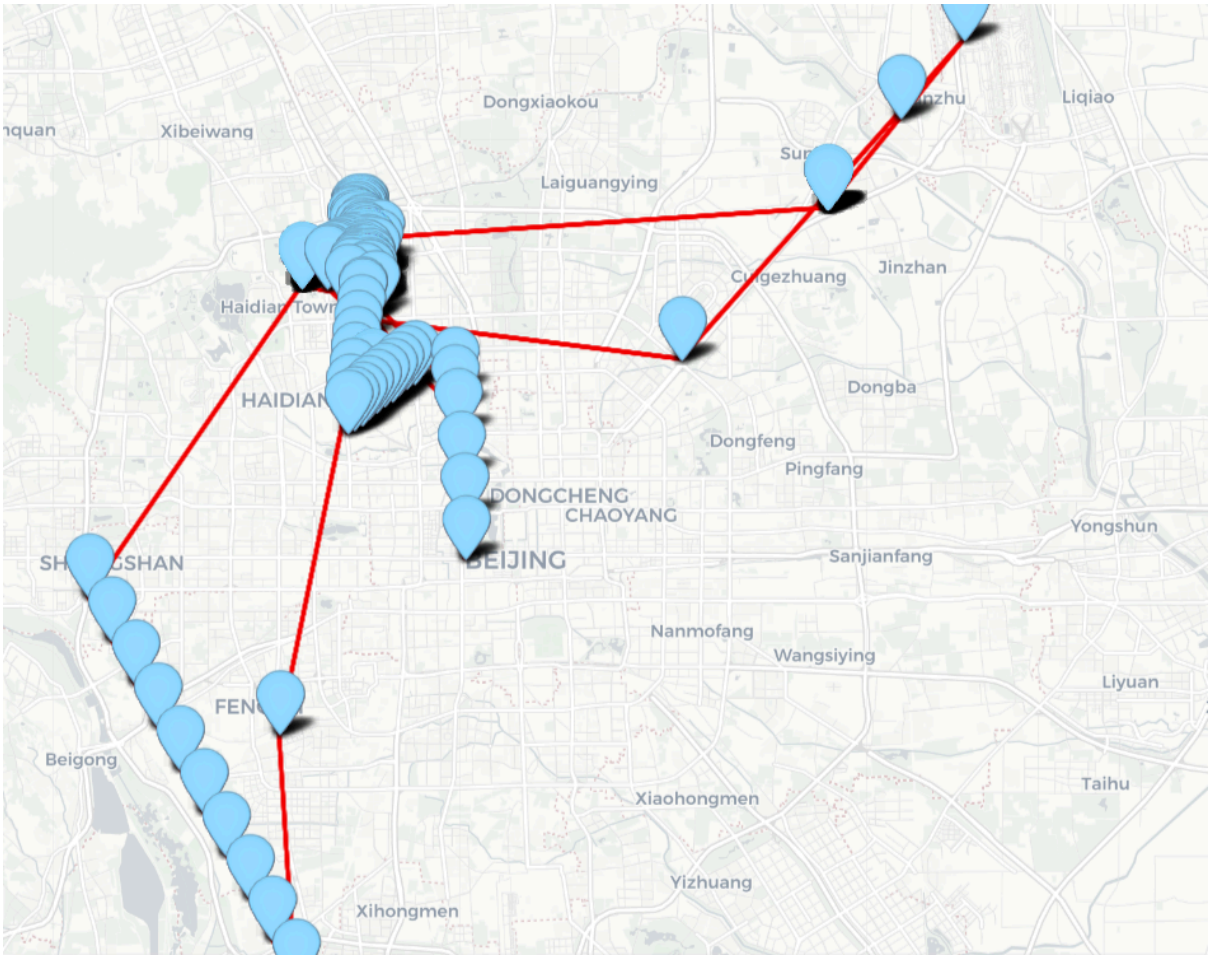
Preprocessing steps for Geolife Dataset

To remove the outliers, which are the stationary users and those who are not active enough, we removed all these users that have less than five location records. To ensure that only common trajectories are considered, we narrowed the range down to only contain the Beijing area. With these operations, the number of users dropped from 180 to 150.

The dataset is also reduced to 48 records for each trajectory with a 30-minute time interval. As the missing points are less and the trajectories are more complete, the interpolation step for this dataset is linear interpolation, that is predicting the missing location assuming the user is on the way at a stable velocity.

As for the discretization here is a grid discretization, which defines a 100×100 grid and considers records in the same grid cell are in the same location.

However, after all the previous operations, the dataset is lacking enough trajectories for the further implementations. To solve this issue, we introduced a method merge, which is combining all the data from the same day. For example, we can combine every Monday together so that we can get active users from all Mondays. This is basically based on our assumption that the users have a similar pattern for each day. And based on the trajectory visualization for a user's Mondays, we can hypothesize that the assumption is considerable for this dataset.



And as we have done for the Shanghai dataset, we are also formatting the processed dataset for the next stage.

2.4 Epic 1: Trajectory Analysis and Reproduction

2.4.1 Problem Definition

The baseline method seeks to replicate the function outlined in 'Trajectory Recovery from Ash: User Privacy Is Not preserved in Aggregated Mobility Data' (Xu et al. 2017). It centers on addressing the trajectory recovery challenge using an unsupervised framework without

pre-existing information. The baseline method comprises three main elements: nighttime recovery, daytime recovery, and cross-day recovery.

The approach is applied to data gathered from both base stations and individual users. The primary challenge is distinguishing whether the mobility records pertain to the same mobile users. As outlined in the referenced paper (Xu et al. 2017), a strategy is introduced for linking trajectories within short time intervals initially, then linking trajectories across different days, ultimately reconstructing the complete trajectory. Successfully connecting the trajectories entails two steps. First, estimate the possibility of next time slot location l_i^{t+1} belongs to a given trajectory based on user mobility pattern, where l_i^{t+1} represents the i_{th} user's location at $t+1$ time stamp. Second, allocate the location of the next time slot to the recovered trajectory that maximizes overall likelihood. The recovered trajectory until time t can be defined as $S^t = [s_1^t, s_2^t, \ldots\ldots, s_N^t]$, where $s_j^t = [q_j^1, q_j^2, \ldots\ldots q_j^t]$ is the j_{th} recovered trajectory until time t and q_j^t represents the recovered location at time slot t. Based on the recovered trajectory, cost matrix $C^t = \{c_{i,j}^t\}_{N*N}$ can be generated, where $c_{i,j}^t$ represents the possibility of linking recovered trajectory s_t with next location l_j^{t+1}. After constructing the cost matrix, the next step involves determining the most suitable assignment of the subsequent time slot location to the recovered trajectory. In decision matrix $X^t = \{x_{i,j}^t\}_{N*N}$, $x_{i,j}^t = 1$ means l_j^{t+1} is linked to the recovered trajectory s_t and $x_{i,j}^t = 0$ means otherwise. The problem of recovering trajectory can be written as:

$$Minimize \sum_{i=1}^{N} \sum_{j=1}^{N} c_{i,j}^t * x_{i,j}^t$$

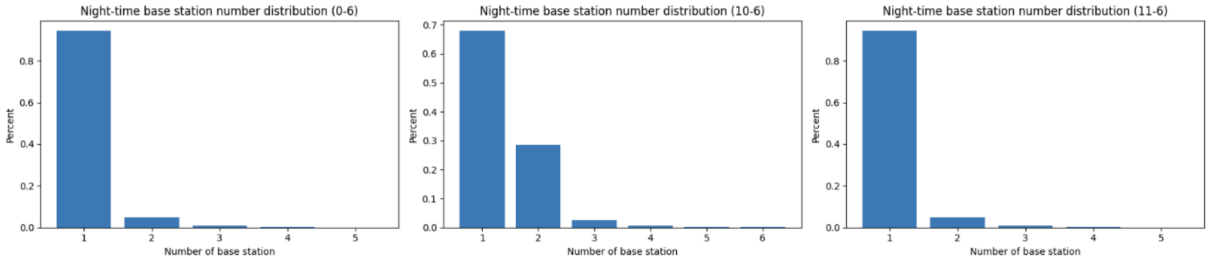
This problem can be solved by applying a Hungarian algorithm. Therefore, the key to successfully recovering trajectories is to construct appropriate cost matrices. In the baseline methods, we follow the reference paper (Xu et al. 2017) to construct three different cost matrices for various time slots, which will be discussed in detail later.

2.4.2 Trajectory recovery

2.4.2.1 Night Time Recovery

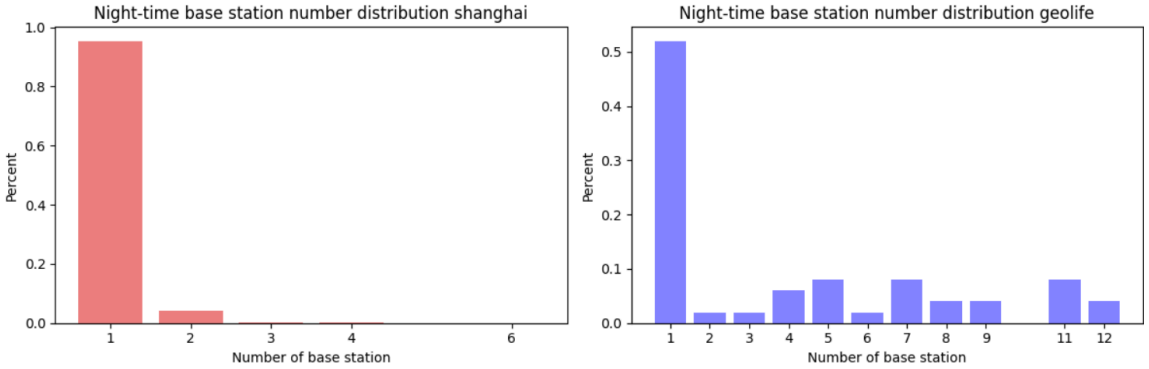
Before the night time recovery, we assume the users tend to maintain stationary during night. We explore the distribution of the number of base stations users visit during the night

to find the appropriate definition of night time.



Distribution of number of base stations visited

The above graph indicates the distribution of the number of base stations visited with different night time slot definitions. Based on the graph, allocating the time slot from 0 am - 6 am results in the highest percentage of people staying stationary. Hence, we define the period 0 am - 6 am as nighttime. We also applied this selection on both shanghai and geolife dataset.



According to the above graph , the Shanghai dataset exhibits a higher percentage of stationary users. This disparity arises due to the inclusion of diverse transportation modes in the GeoLife dataset. Consequently, occupations like taxi drivers, who operate primarily at night, are accounted for, resulting in a reduced percentage.

Since most users remain stationary during the night time, we construct our cost matrix based on this principle. We predict the location in the next time slot remains unchanged. The predicted next time slot location is denoted as $\hat{l}_i^{t+1}$ and it can be derived through formula $\hat{l}_i^{t+1} = q_i^t$. The distance between predicted next location $\hat{l}_i^{t+1}$ and actual nex location l_i^{t+1} is used as the cost element $c_{i,j}^t$ in the cost matrix, which represents the cost to link the predicted next location to the recovered trajectory s_i^t. The Hungarian algorithm can be utilized with the cost matrix to connect the predicted next location to the reconstructed trajectory, facilitating the recovery of the entire trajectory during nighttime.

2.4.2.2 Day Time Recovery

Usually users share a different moving pattern during the day time compared with the night time. Regarding the continuous moving during the day time, We assume that users maintain a constant speed and trajectory direction within the given time intervals. Based on assumption, the predicted next location $\hat{l}_i^{t+1}$ can be calculated by using the formula: $\hat{l}_i^{t+1} = q_i^t$

+q_i^t-q_i^{t-1}, where q_i^t represents the location at time slot t and q_i^{t-1} represents the location at time slot $t-1$ within the same i_{th} trajectory. q_i^t-q_i^{t-1} represents the distance and direction user moved from timestamp $t-1$ to t. We use the distance between the predicted next location and actual next location as the cost matrix and the formula can be written as $c_{i,j}^t = distance(\hat{l}_i^{t+1}, l_j^{t+1})$. We converted the latitude and longitude coordinates from our dataset into Cartesian coordinates before calculating the Euclidean distance. We use the tangent plane projection method (Buchholz et al. 2022) to transfer the longitude and latitude.

$$x = 111319.44 * cos(lat_0) * (lon - lon_0)$$

$$y = 111319.44 * (lat - lat_0)$$

where lat_0 and lon_0 refer to the latitude and longitude of reference point respectively. We choose the median of the latitude and longitude of the set of unique tokens as the reference point. After getting x, y on the Cartesian axis, we can apply the euclidean distance formula. Applying the Hungarian algorithm can help us to recover the trajectory during the night time. For the initial points, we incorporated some nighttime data to aid in predicting daytime points, thereby automatically linking the nighttime trajectory with the daytime trajectory.

2.4.2.3 Cross Day Recovery

Upon reconstructing both the nighttime and daytime trajectories, we can generate the user's trajectory for a given day. Expanding the time frame to recover the user's trajectory requires incorporating additional assumptions into the baseline method. We assume the trajectories of the same user between consecutive days share the similar pattern and there is a huge difference in trajectory pattern between different users. Based on these two assumptions, information gain is applied to measure the similarity between two trajectories. To calculate the information gain between different trajectories, we first calculate the entropy of a single trajectory. The recovered trajectory for i_{th} user in d_{th} day is denoted as U_i^d. The entropy formula is listed below:

$$H(U_i^d) = -\sum_{k=1}^{n} \frac{f_k}{\sum_{k=1}^{n} f_k} log(\frac{f_k}{\sum_{k=1}^{n} f_k})$$

In this formula, f_k is the frequency of the k_{th} base station visited by the user within a given trajectory. After calculating the entropy for each trajectory, we can calculate the information gain between different trajectories and use it as a measure of the similarity between different trajectories. Since in this project, we focus on the recovering the trajectory within consecutive time period, the information gain for trajectory within consecutive two days is denoted as $G(U_i^d, U_j^{d+1})$ and the formula can be written as

$$G(U_i^d, U_j^{d+1}) = H(U_i^d + U_j^{d+1}) - \frac{H(U_i^d)+H(U_j^{d+1})}{2}$$

where the $H(U_i^d, U_j^{d+1})$ can be generated by combining two-day trajectory together and calculating the entropy of the combined trajectory. A near-zero information gain indicates high similarity between the selected two trajectories, while a value close to 1 suggests the

opposite. Therefore, the information gain can be used to construct the cost matrix for cross day, where $c_{i,j}^{d} = G(U_i^d, U_j^{d+1})$. At this stage, the Hungarian algorithm will be employed to merge trajectories with the highest similarity, iteratively reconstructing predicted trajectories for all users.

2.4.2.4 Match predicted and actual trajectory

Applying the above three stages, we can predict the whole trajectories for all users. To facilitate correct evaluation, the predicted trajectories should be paired with the actual trajectories. After applying the algorithm, as the predicted trajectories' IDs are masked, we must execute a greedy matching process to retrieve the IDs and match the predicted trajectories with the actual ones. The process involves selecting one recovered trajectory from the predicted set and comparing its similarity to all trajectories in the actual set. The similarity metric used in the greedy matching is the percentage of exact matching points across the entire trajectory. Once all similarities are obtained, we pair the predicted trajectory with the one in the actual set with the highest similarity, and then remove both trajectories from their respective sets. After pairing all trajectories, the evaluation can be implemented.

2.4.3 Evaluation(Baseline Model)

Following the algorithm our team designed to recover the trajectory of each user, we need to assess how well our model performs, which for the specific project, requires us to design metrics to evaluate how accurately the model predicts each user, how close are we from the location points we predicted compared with ground truth, and more importantly, remember the goal of the project? Our project aims to recover user trajectories from ashes and acquire their personal information consequently. Hence, the last metric will focus on quantifying how good the model is to crack the code based on each user's trajectory.

2.4.3.1 Evaluation Metrics Introduction

Given the illustrated aspects we would like to validate and quantify from the outcome. We designed three metrics - accuracy, error distance, and uniqueness.

1) Accuracy:

   Given the i-th user's authentic trajectory: $T_i(\text{actual}) = [a_1, a_2, a_3 \dots a_T]$ and the predicted trajectory generated from our model: $T_i(\text{predicted}) = [p_1, p_2, p_3 \dots p_T]$.
   Each a_k/p_k denotes the current user's location point (latitude, longitude) at the time t (t spans between 1 and 48(timeslots) x M(days of the whole period)), and i ranges from 1 to N(total number of unique users)
   For the first metric, the one provided in the reference paper is:

$$A = \frac{1}{N} \sum_{i=1}^{N} \frac{|Ti(actual) \cap Ti(predicted)|}{|Ti(actual)|}$$

   While our accuracy formula is:

$$A = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{t=1}^{T=48*M} count(i)}{|Ti(actual)|}$$

$$(\text{count}(i) \Leftarrow ai == pi\ ?\ 1 : 0)$$

To better demonstrate the issued formulas, here is a simple representation:

User1's trajectory	Actual		Predicted	
Timestamp	**Latitude**	**Longitude**	**Latitude**	**Longitude**
1	34.342	102.778	35.372	107.231
2	34.681	103.332	34.681	103.332
3	35.372	107.231	34.342	102.778

In the accuracy defined in the original paper, they used the intersection set of actual location points and predicted one to represent correctly predicted points, while in our method, we count the number of matched latitude and longitude pairs between predicted and actual location points. Hence, in the graph shown above, we have (34.342, 102.778), (34.681, 103.332) and (35.372,107.231) existing in both location sets, but we have only one matched pair (34.681, 103.332). Hence, the results are:

$$A(\text{original paper}) = \frac{3}{3}$$

$$A(\text{new}) = \frac{1}{3}$$

We believe that the matched pairs following the timestamps outperform in representing recovery accuracy since in the model we are predicting each user's next location following a daily time frame. Suppose we want to investigate Tom's trajectory, given the actual trajectory being Area1 -> Area2 while the predicted trajectory is Area2 -> Area1, it is logically inconceivable to say that we correctly predict the current user's route.

In practice, we merge the actual and predicted trajectory data on user id and then we calculate the result with the count of the number of rows that share the same latitude and longitude divided by the total number of rows.

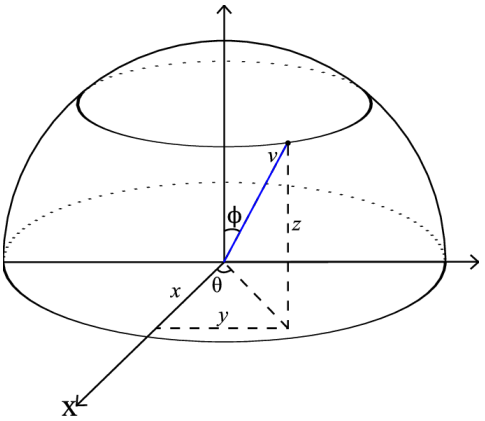
2) Error Distance:

The former metric gives us a sense of how great the model performs and works as an upper bound, so we also need a lower bound to quantify how bad the model can be. In this case, we need to introduce another metric - error distance to calculate the distance between the predicted location points and actual location points.

Intuitively, we use the Euclidean distance to calculate the range between two points:

$$\text{Euclidean Distance } (P_1, P_2) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

but we also should notice that the formula works correctly on the Cartesian axis while the location points we have are from Earth which is a large Spherical axis. So, we will need to project the location points from the spherical axis on the Cartesian axis first:



Given the point on Earth, we can project it on the Cartesian axis:

$$x = R * \sin(\phi) * \cos(\theta)$$
$$y = R * \sin(\phi) * \sin(\theta)$$
$$z = R * \cos(\phi)$$

(with R being the radius of Earth, which is about 6371 km)

Also, we have converting equations between latitude longitude and ϕ(the polar angle) and θ(the azimuthal angle):

$$\phi = 90^\circ - lat$$
$$\theta = long$$

Moreover, since user data is collected within one city, we can ignore the height difference. Therefore, the equation to project latitude and longitude onto the Cartesian axis is:

$$x = R * \sin(90^\circ - lat) * \cos(long)$$
$$= R * \cos(lat) * \cos(long)$$

$$y = R * \sin(90^\circ - lat) * \sin(long)$$
$$= R * \cos(lat) * \sin(long)$$

Given the merged table containing actual and predicted trajectories, we will apply the converting function on location points from rows where the actual and predicted pairs do not match. Then we apply the Euclidean distance function to calculate the distance for each row and get the average distance as the result.

3) Uniqueness:
Given we have acquired an accuracy of 30% with an error distance of 10 km, we can read from these figures that we guessed about 30 location points correctly and we are about average 10 km far from ground truth for location points that do not match.

However, given the merged table snippet of 5 users in one day like this:

User ID	Date	Location Point (latitude, longitude)
2fa838f05a4561592741564645a55f14	1	A
3f8214d9d42138dc4f5219bb36d99779	1	A
40d41c461cd11402231d8e63b1d57314	1	A
dfaa9308e73b344810ca7cadf55006a3	1	B
3a0e916fc105e88d43da43c2402dbd30	1	D

Without further information provided for each user, it is impossible to guess each user's identity except for users "dfaa9308e73b344810ca7cadf55006a3" and "3a0e916fc105e88d43da43c2402dbd30", since we can see that of all the users, only these two user's share different location points and "different" implies "unique". Now given we gathered information for each location point shown above that A - Apartment A, B - House A, and C - House B, it is fairly easy to say that we know "dfaa9308e73b344810ca7cadf55006a3" is the user who lives in House A and "3a0e916fc105e88d43da43c2402dbd30" is another person living in House. However, even if we know where location point A is located, we cannot confirm the other three users' identities, and we require further location points for the three users provided.

Notice also in the above example that we picked a random day while what we want to do is inspect a user's chronic behavior since there is a huge difference between "I live in Apartment A" and "I 've been to Apartment A once". Hence, given a dataset of a period, we will first convert it to aggregate all location points belonging to each user and select the top 5 frequent locations which denotes each user's most visited places, which will make it easier to validate each user's personal information since these places infer to places where they live, work and other places of interest.

In conclusion, we will aggregate and select the top 5 locations first and then perform uniqueness checking for each top n locations of all users:

User ID	Location Point (latitude, longitude)	Frequency
1st Frequent Location		

2fa838f05a4561592741564645a55f14	A	30
3f8214d9d42138dc4f5219bb36d99779	A	30
40d41c461cd11402231d8e63b1d57314	B	30
dfaa9308e73b344810ca7cadf55006a3	B	30
3a0e916fc105e88d43da43c2402dbd30	C	30
2nd Frequent Location		
2fa838f05a4561592741564645a55f14	D	20
3f8214d9d42138dc4f5219bb36d99779	D	18
40d41c461cd11402231d8e63b1d57314	E	22
dfaa9308e73b344810ca7cadf55006a3	F	15
3a0e916fc105e88d43da43c2402dbd30	G	13
3rd Frequent Location		
...		

Now, we can perform uniqueness checking using:

$$\text{Uniqueness} = \frac{Count(unique\ users)}{Count(total\ users)}$$

(count(unique users) += frequency(location point per user) == 1 ? 1 : 0)

For the top-1 location, we inspect that only user location point C appeared once, which means after revealing all ground truth information about top 1 locations A, B, and C, we are only able to confirm the identity of user ”3a0e916fc105e88d43da43c2402dbd30”. Given one more point, with 2 location points provided, we can validate the identity of two more users - “40d41c461cd11402231d8e63b1d57314" and “dfaa9308e73b344810ca7cadf55006a3” since they share different snd frequent locations, you can imagine that user A and B living in the same apartment but they work at different companies.

So the uniqueness figure will be:

$$\text{Uniqueness(top 1)} = \frac{1}{5}$$

$$\text{Uniqueness(top 2)} = \frac{3}{5}$$

…

2.4.3.2 Evaluation Process

(1) After the matching process, we paired actual trajectory with the predicted trajectory for each user:

Actual trajectory:

```
expected_df.head()
```

	uid	t	lat	long	date
0	2fa838f05a4561592741564645a55f14	0	31.337188	121.409143	6-18
1	2fa838f05a4561592741564645a55f14	1	31.316339	121.302556	6-18
2	2fa838f05a4561592741564645a55f14	2	31.316339	121.302556	6-18
3	2fa838f05a4561592741564645a55f14	3	31.316339	121.302556	6-18
4	2fa838f05a4561592741564645a55f14	4	31.202929	121.443709	6-18

Predicted trajectory:

```
actual_df.head()
```

	uid	t	lat	long	date
0	2fa838f05a4561592741564645a55f14	0	31.243355	121.414892	6-18
1	2fa838f05a4561592741564645a55f14	1	31.243355	121.414892	6-18
2	2fa838f05a4561592741564645a55f14	2	31.243355	121.414892	6-18
3	2fa838f05a4561592741564645a55f14	3	31.243355	121.414892	6-18
4	2fa838f05a4561592741564645a55f14	4	31.243355	121.414892	6-18

(2) In the next step, we merged two tables based on user ID, timestamp, and date. Then, we generate an extra row “equal” of which the value will be set to 1 if the actual location matches the predicted location:

	uid	t	lat_x	long_x	date	lat_y	long_y	equal
0	2fa838f05a4561592741564645a55f14	0	31.337188	121.409143	6-18	31.243355	121.414892	0
1	2fa838f05a4561592741564645a55f14	1	31.316339	121.302556	6-18	31.243355	121.414892	0
2	2fa838f05a4561592741564645a55f14	2	31.316339	121.302556	6-18	31.243355	121.414892	0
3	2fa838f05a4561592741564645a55f14	3	31.316339	121.302556	6-18	31.243355	121.414892	0
4	2fa838f05a4561592741564645a55f14	4	31.202929	121.443709	6-18	31.243355	121.414892	0
...	...	...	...	...	...	...	...	...
19435	3a0e916fc105e88d43da43c2402dbd30	43	31.258428	121.369522	7-2	31.337188	121.409143	0
19436	3a0e916fc105e88d43da43c2402dbd30	44	31.258428	121.369522	7-2	31.337188	121.409143	0
19437	3a0e916fc105e88d43da43c2402dbd30	45	31.258428	121.369522	7-2	31.337188	121.409143	0
19438	3a0e916fc105e88d43da43c2402dbd30	46	31.147427	121.355113	7-2	31.337188	121.409143	0
19439	3a0e916fc105e88d43da43c2402dbd30	47	31.147427	121.355113	7-2	31.337188	121.409143	0

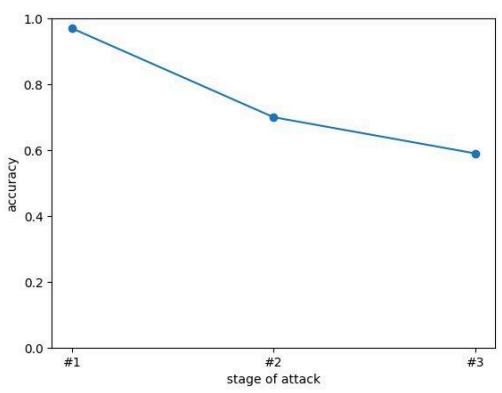
(3) For accuracy calculation, we simply sum up the values in the equal column and then divide this number by the total number of rows

(4) For error distance, we will apply two functions on each row, the first one is used to project latitude and longitude on the Cartesian axis and the second one is used to calculate the Euclidean distance for two pairs of x and y. Then we will apply average(distance[...]) to get the average error distance.

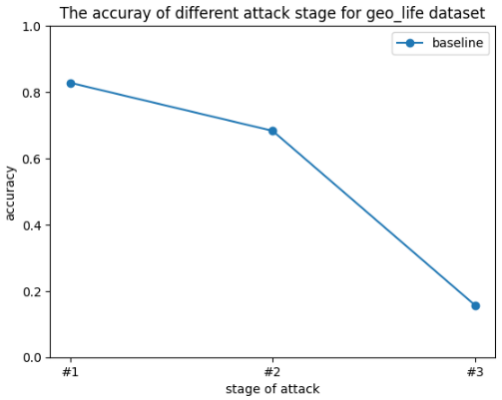
(5) For uniqueness, we aggregated location points based on each user's user ID, then counted the frequency for each location point of each user, sorted in descending order, kept the top 5 location points, and added an extra column "top-n" with a value ranging between 1 and 5 denoting the rank of the frequent place. In the next step, we calculated the number of unique location points whose frequency is equal to 1 and divided the number of unique users by the total number of distinct users. We repeatedly compute for each top-n value so that we have top-5 uniqueness values.

2.4.3.3 Evaluation Result

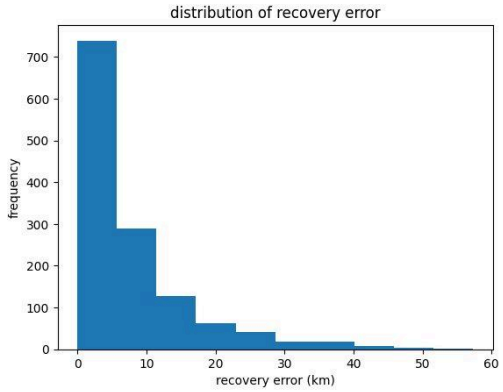
In the evaluation part, we test the accuracy, error distance and uniqueness for shanghai and geolife dataset. For the Shanghai dataset we test the consecutive 15 day trajectories and for geolife we test the consecutive 7 days.



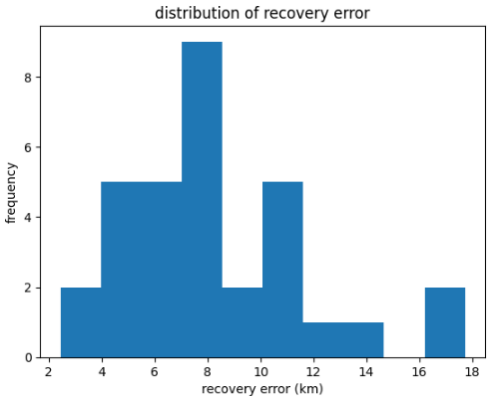
Accuracy for shanghai



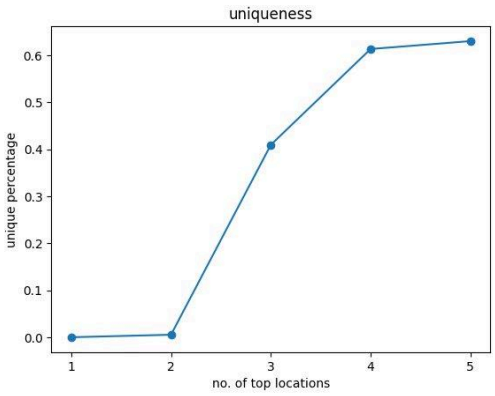
Accuracy for geolife



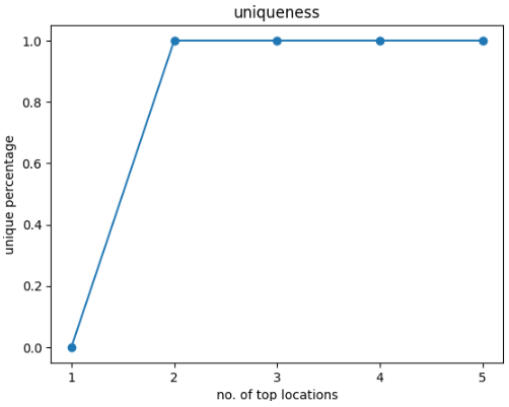
Error distance for shanghai



Error distance for geolife



Uniqueness for shanghai



Uniqueness for geolife

2.4.3.4 Conclusion From Results

From the result shown above, we can see that the baseline model can capture the nighttime pattern of users so we have over 80% accuracy in predicting the nighttime trajectories, while in the daytime when people are moving, the model performs poorer since we coarsely predict current user's next location point based on a constant velocity while user speed changes in practice. Also, the naively implemented greedy method for user ID pairing even reduced cross-day recovery accuracy and increased error distance.

The next thing that needs to be noticed is the uniqueness, it is not hard to find that the overall uniqueness of the Geolife dataset is greater than that of the Shanghai dataset. The main reason is that in the Shanghai dataset, user data are collected via the base station that they are located at while in the Geolife, the location points are each user's accurate ones. So, although user1 and user2 in Shanghai might live in different apartments and work at different companies, the trajectory we got could be Area A -> Area B given two apartments both reside in Area A and two companies reside in Area B.

One more thing to note is that there is a large variance in the model performance. In the Shanghai dataset, we reached a minimum of 60% cross-day trajectory accuracy while in the Geolife we could only reach about 20%, implying that the model has not been resilient enough for variant environments.

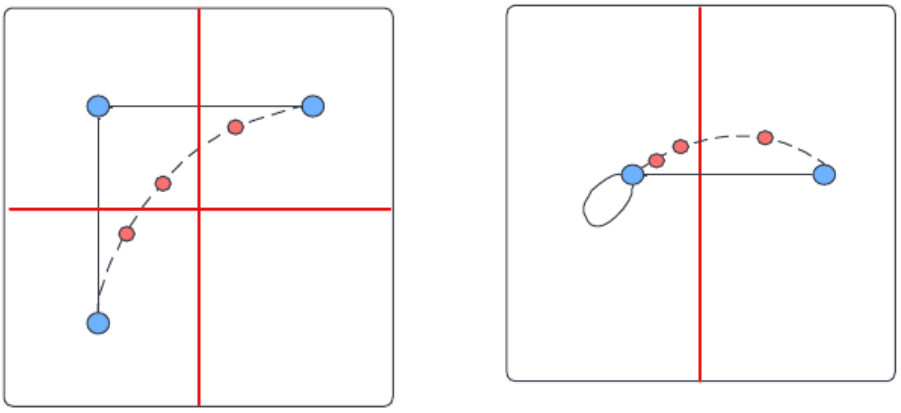
In conclusion, we managed to develop a baseline model that could at least predict 20% of users' trajectories while guaranteeing most of the location points we predicted are within 20 km far from actual location points. Moreover, we could confirm at least 60% of each user's identity given the top 5 location information provided. However, we need a more resilient model to perform stably in variant environments.

2.5 Epic 2: Deep Learning Model for Enhanced Attack Efficacy

The paper (Xu et al. 2017) we are reproducing was published over six years ago. Since then, the field of deep learning has been advancing rapidly. In particular, sequence models such as recurrent neural networks and transformers (Vaswani et al. 2017). In addition, there is an increasing number of open source trajectory datasets made available online. There is a potential risk of user re-identification from open source aggregated datasets through the use of deep learning, as it have been demonstrated in another study (Buchholz et al. 2022) that demonstrated the use of neural networks in reconstructing trajectories perturbed by differential protection mechanisms.

In this section, we will simulate a scenario where an attacker has access to a trajectory dataset with similar distribution to our target aggregated location data. We will explore the use of supervised learning and neural networks in trajectory recovery from aggregated location data.

One key advantage of neural networks in trajectory modeling is the ability to account for non-Markovian dynamics in observed trajectories. 2 examples of non-Markovian dynamics are provided in the below figure. Blue markers denote discretized locations (base stations); solid lines denote observed trajectory; red markers denote unknown actual locations at the sampled time slot; the dotted lines denote the unknown actual trajectory; and the red line denotes the discretization boundaries.



Examples of non-Markovian dynamics.
Left: zig-zag patterns
Right: Time-lag

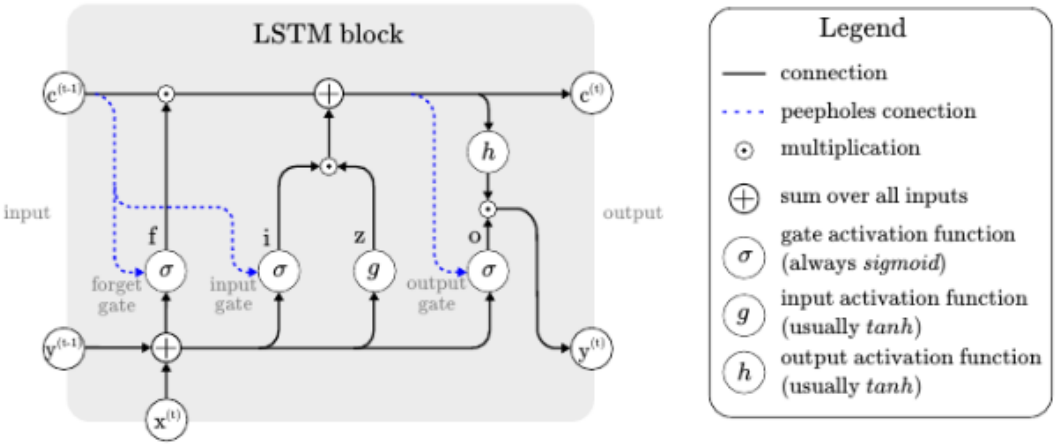
Another advantage is the ability of supervised methods to learn localized patterns (e.g. peak hours, traffic routes, and travel habits of the local population). While such patterns learned by the model may not generalize to datasets with different distributions (i.e. in different

locations and time periods), supervised methods generally out-performs unsupervised ones on datasets with similar distribution to the training set.

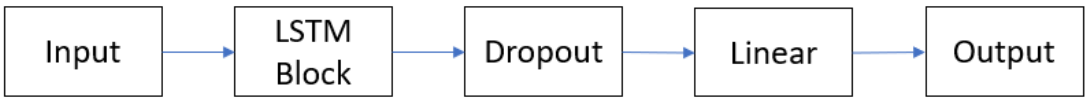
2.5.1 LSTM model

2.5.1.1 Methodology

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that has cyclic connections making it powerful for modeling sequences (Sak, Senior, & Beaufays, 2014). For our trajectory recovery task, trajectories are defined as points that are related to the sequence of coordinates, for the testing condition, the relationship of these points is unknown. LSTM could potentially offer valuable insight into the succession of points that we can use to link consecutive points together. The input gate and forget gate of LSTM also act as a feature selection process to identify potential relationships between points.



The architecture of vanilla LSTM block (Van Houdt, Mosquera, & Nápoles, 2020)



The architecture of our model

The input layer contains the trajectory data with shape (batch_size, sequence_length, xy coordinates). The input data will be normalized to (-1,1) for better comparison because the difference in latitude and longitude is small in their original value which will cause the problem of mode collapse.

The LSTM will then process the input sequence and generate the sequence of hidden states. The parameter we used is with 2 layers of LSTM and 40 features within the block. Dropout with 0.5 is then applied to avoid overfitting and improve model generalization ability. The Linear layer will then map the input tensor with a linear transformation to the output space.

The output layer of the model will be the predicted coordinates for the next timestamp.

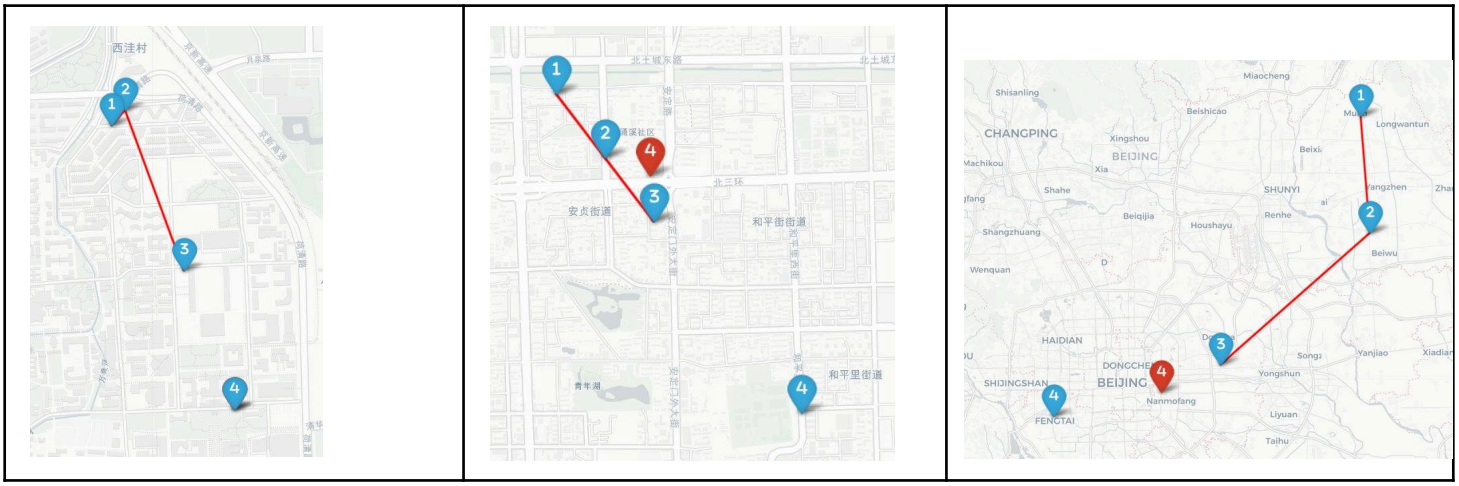
2.5.1.2 Data processing

To further process the input data for LSTM, we need to normalize the data with the center point of our dataset to transform both latitude and longitude into a range of -1 to 1. Then we collect features of 3 consecutive coordinates and the next coordinate as the target using the sliding window approach. By iterating through all data, we retrieve data for training and testing the LSTM.

2.5.1.3 Trajectory recovery from test data

After the model is trained, we test the model on an aggregated dataset to predict the trajectory. For the first 3 points, we just assume all users stay stationary and append the points that have the closest Euclidean distance to their last point of the trajectory. After having 3 points, the trajectories are then fed into the LSTM to obtain the prediction of the next point. The coordinates for the next timestamp are distributed to each user using a greedy approach where it assigns to the trajectories that have the closest predicted next location. The input for LSTM will now be the last 3 coordinates of each trajectory. This process is repeated until 48 points are collected. A graphical illustration is as below:

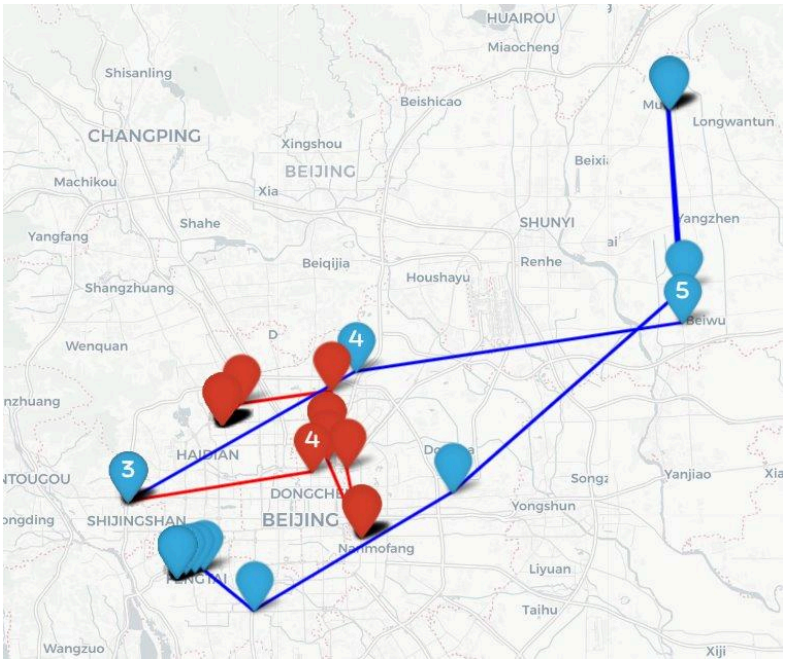
2.5.1.4 Results and discussion



Single-step prediction of the LSTM model

The above graphs show the prediction of LSTM in one step where linked blue dots are observed. Number 4 with the blue color is the ground truth of the next step and the red one is the predicted location. The first graph is a successful prediction while the other 2 predictions lie in the general direction of the ground truth.

For the Shanghai dataset, most predictions lie in the same spot for the whole day due to the relatively larger range of base stations and this behavior is not desired.



Prediction of the whole day

The LSTM method relies on correct prediction for each point for subsequent predictions in testing data where all user id is masked. That means one wrong prediction may drive the whole trajectory of the original trajectory. The above result is a prediction for 48 points throughout the day and the performance is not satisfactory.

The metric results obtained from LSTM are as follows:

Data	Accuracy (Pointwise)	Average Recovery Error	Uniqueness
Shanghai 1 Day	Night: 0.404 Day: 0.117	16.72(km)	1: 0.0, 2: 0.213, 3: 0.361, 4: 0.365, 5: 0.365
Geolife 1 Day	Night: 0.311 Day: 0.152	9.614(km)	1: 0.818, 2: 0.818, 3: 0.818, 4: 0.818, 5: 0.818
Shanghai 15 Days	Night:0.195 Day:0.019	21.468(km)	1: 0.0, 2: 0.167, 3: 0.525, 4: 0.61, 5: 0.616

The metric performance above shows the limitation of LSTM in trajectory recovery in this task. Firstly, the low uniqueness shows that many predicted trajectories are identical. With further inspection, the predicted trajectories tend to stay in the same spot for the whole period. This situation is most critical in the Shanghai dataset where the amount of users is huge and the number of base stations is low. The LSTM algorithm will always predict a point close to the stationary point and the flow of people cannot be predicted because it considers each trajectory independently. Another evidence for this phenomenon is that the accuracy at night time is much higher because users tend to stay stationary at night time which is in line with the model's prediction. Therefore, we conclude that LSTM might not be the appropriate

method for the given dataset and task due to a large number of interfering points in the trajectory recovery process. Also, the sparse nature of the base station limits the trajectory observed to only a few points for a day which makes it difficult for LSTM to predict the next location. Finally, the sampling rate of 30 minutes will result in sharp turns in the trajectory that will also influence the recovery process.

2.5.2 Neural Trajectory Modeling

2.5.2.1 Mathematical Motivation

The average Euclidean distance objective used in step 1 and step 2 in 'Trajectory Recovery From Ash' (Xu et al. 2017) can be interpreted as a negative log likelihood loss. If we assume the square root of the Euclidean distance between a user and a base station is normally distributed (with $\mu = 0$ and $\sigma^2 = 1$), we arrive at the maximization of joint likelihood after a series of derivations.

$$\begin{aligned}
\min \sum_{(i,j)\in\{matched\}} \|\hat{l}_i^{(t)} - l_j^{(t)}\| &= \min \sum_{(i,j)\in\{matched\}} \frac{\left(\sqrt{\|\hat{l}_i^{(t)} - l_j^{(t)}\|} - \mu\right)^2}{2\sigma^2} \\
&= \max \sum_{(i,j)\in\{matched\}} \log \exp \frac{\left(\sqrt{\|\hat{l}_i^{(t)} - l_j^{(t)}\|} - \mu\right)^2}{-2\sigma^2} \\
&= \max \prod_{(i,j)\in\{matched\}} \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{\left(\sqrt{\|\hat{l}_i^{(t)} - l_j^{(t)}\|} - \mu\right)^2}{-2\sigma^2} \\
&= \max \prod_{(i,j)\in\{matched\}} \Pr\left(l_j^{(t)} | \hat{l}_i^{(t)}\right)
\end{aligned}$$

Likelihood maximization objective used to find similar points belonging to a user

Where $\{matched\}$ denotes the set of 2-tuples matched by the Hungarian algorithm.

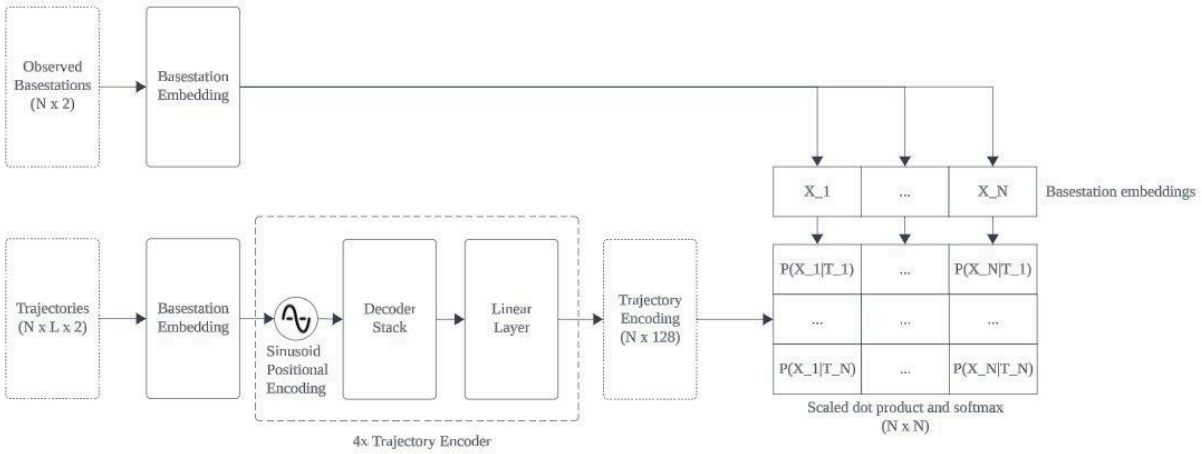
2.5.2.2 Methodology

Equipped with the probabilistic interpretation, a direct evolution of 'Trajectory Recovery From Ash' (Xu et al. 2017) is to use a deep neural network to model $\Pr(l_j^{(t)})$.

A trajectory consists of a sequence of discrete tokens (base stations). With this similarity with language modeling in mind, we propose a transformer based architecture to model $\Pr(l_j^{(t)} | l_i^{(1)}, \ldots, l_i^{(t-1)})$.

But different from language modeling, we are met with additional constraints in which the available pool of tokens varies at different time steps. For example, in a 2 users scenario: user 1 travel from base station A to base station B while user 2 stayed at C; this lead to the set of observed tokens changes from $\{A,\ C\}$ to $\{B,\ C\}$.

We require the predicted probability distribution of tokens to be consistent with the observation at all time (i.e. $P(A) = 0$ at the next time step). Hence, we adapted the non-parametric image classification architecture in CLIP (Radford et al. 2021) for predicting likelihood of tokens at inference.



Overall Architecture of our model

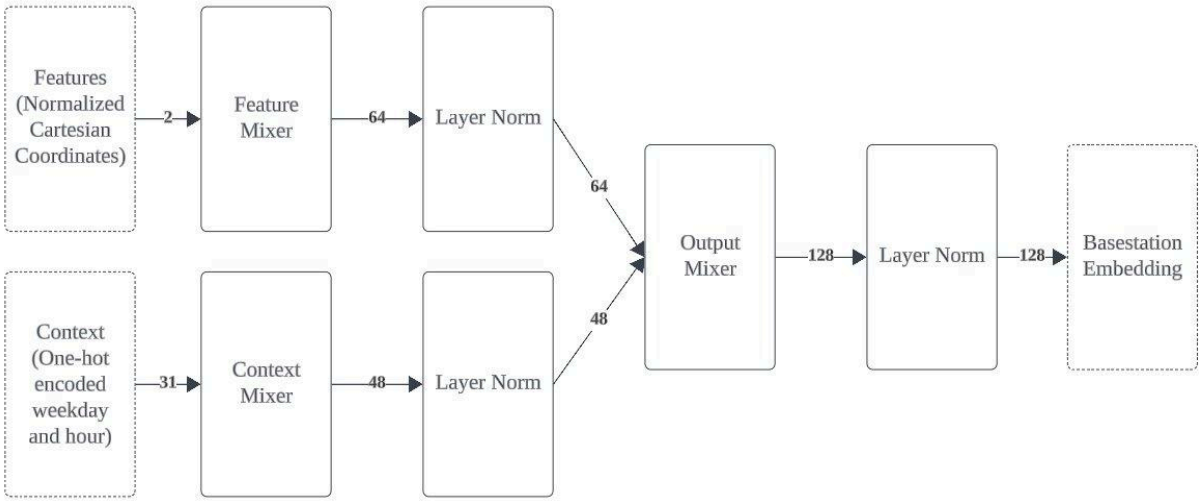
Given a trajectory and a list of observed base stations, our model first computes the embedding of the trajectory and the embedding for each observation using the base station embedding model which we will introduce later.

Then we prepend a special `[BOS]` (begin of sequence) token to the beginning of the embedded sequence. The `[BOS]` token is a learned tensor. Probabilistically speaking, it can be interpreted as the prior $\Pr(l_j^{(t=1)})$.

Trajectory tensor with [BOS] prepended

To encode the trajectory as a vector, we first apply sinusoid positional encoding (Vaswani et al. 2017) to the trajectory tensor. Followed by a stack of 4 encoders with causal attention masks such that the all time steps can be trained parallelly.

The last token of the transformer output is used to compute the logits of the distribution. We pass it through an additional linear layer, then compute dot-product with the list of base station embeddings we computed earlier to obtain the logits. We can use softmax activation to convert the logits into a probability distribution.



Architecture of the base station embedding model,
number on edges indicates dimensionality of the layer output

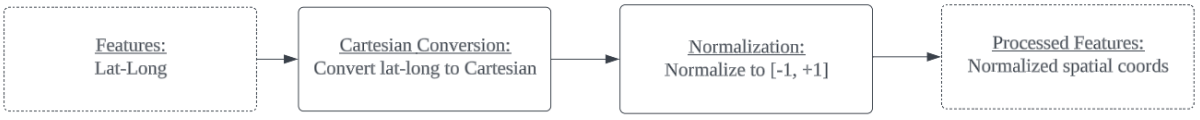
For the base station embedding model, ideally we would like it to have spatiotemporal locality (i.e. nearby locations should have similar statistical properties) while being able to model non-stationary human behaviors.

We adapted the encoder of RAoPT (Buchholz et al. 2022) and added layer norms after each mixer block. The feature mixer takes spatial coordinates of a token (2-dimensional) as input, while the context mixer takes a concatenation of the one-hot encoded hour (24-dimensional) and the one-hot encoded week day (7 dimensional) as input. The output is a 128-dimensional embedding vector.

The architecture has built-in spatial locality but not temporal locality. And we assume the behavior of users to be weekly periodic only and there is no behavioral drift in time.

2.5.2.3 Data Pre-processing

In addition to the pre-processing methodology listed in Section 2.3.3.1, we performed the following steps during training and inference of the model.



Feature pre-processing steps

Cartesian conversion:
The latitude and longitude of each token is converted to Cartesian coordinates using tangent plane projection method in (Buchholz et al. 2022).

$$x = 111\,319.44 * \cos lat_0 * (lon - lon_0)$$
$$y = 111\,319.44 * (lat - lat_0)$$

Mathematical definition of the Cartesian conversion:
lat_0 is latitude of the reference point, lon_0 is the longitude of the reference point

We choose the median of the latitude and longitude of the set of unique tokens as the reference point.

Normalization:
We apply the below formula to normalize the converted coordinates into a range of [-1, +1].

$$x_{norm} = norm(x) = 2(\frac{x - x_{min}}{x_{max} - x_{min}}) - 1$$

2.5.2.4 Training

2.5.2.4.1 Overview

We train the model using self-supervised next token prediction similar to language modeling. The model is tasked to predict a probability distribution $\Pr(l_j^{(t)} | l_i^{(1)}, ..., l_i^{(t-1)})$ given the ground truth at previous steps $l_i^{(1)}, ..., l_i^{(t-1)}$ for all $t > 1$. We minimize the cross-entropy loss between the predicted distribution and the ground truth (a one-hot categorical distribution).

2.5.2.4.2 Hyper-parameters

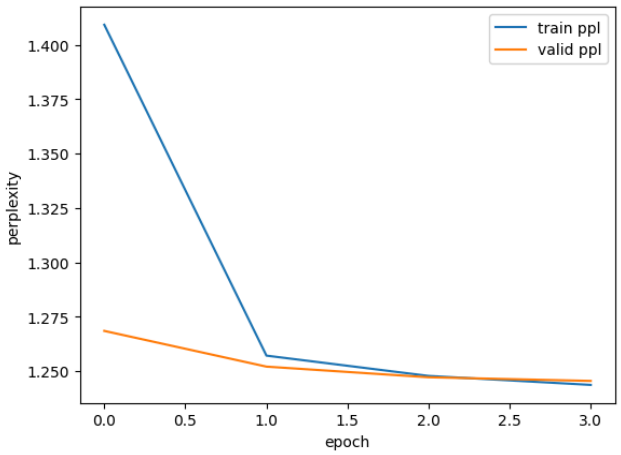
We train the model using trajectories not in the test set used in baseline evaluation. We split the trajectories into a 80% training set and a 20% validation set. All results reported in the next section are from the validation set.

We used the Adam optimizer with default parameters, the learning rate decays with a factor of 0.5 after every epoch, and a dropout probability of 0.2. The model is trained on GPU over 5 epochs.

2.5.2.4.3 Results

We trained the model on the processed Shanghai Telecom dataset discretized with 100 clusters and 30 minute time slots.

Model dimension	Perplexity	Mean Error (km)	Accuracy
16	1.32	1.451	0.9459
32	1.27	1.060	0.9460
64	1.25	0.780	0.9462
128	**1.25**	**0.725**	**0.9464**



Evolution of the perplexity of the best model (128-dimensional) up to its best epoch

The 128-dimensional model is the best performing one, and will be used for subsequent analysis. The parameters of the 128-dimensional model are available on the model checkpoint on Github (`src/ml/checkpoints/sh30-c100-best.pt`).

2.5.2.5 Trajectory Recovery

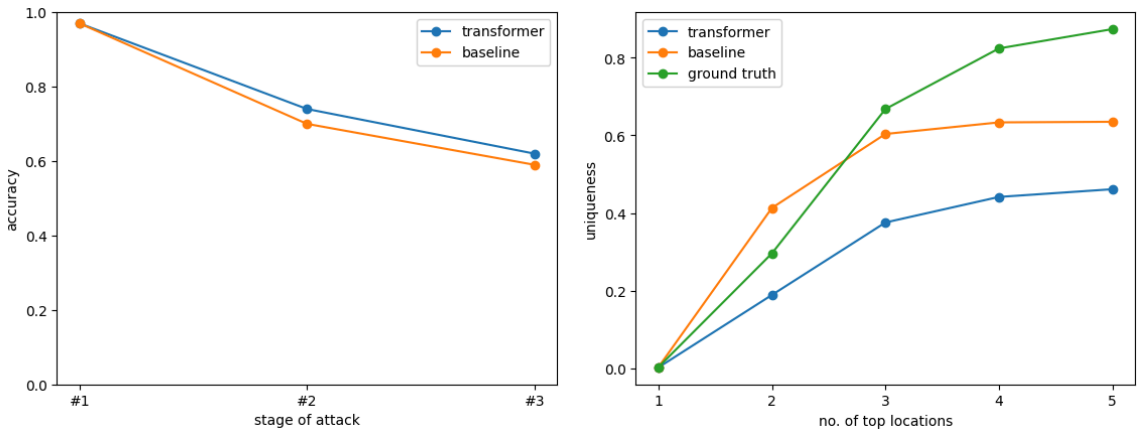
2.5.2.5.1 Overview

During inference time, we use the trained model to compute the negative log-likelihood $-\log \Pr(l_j^{(t)} | l_i^{(1)}, \ldots, l_i^{(t-1)})$ of all pairs of trajectories and base stations. Then we perform bipartite matching to minimize the sum of negative log-likelihood to achieve the goal of joint likelihood maximization. Finally, we append the matched base stations to each trajectory and predict the next point in time until we obtain the trajectory of an entire day.

Essentially, we replace the heuristic cost in night time and day time proposed by (Xu et al. 2017) with a neural likelihood. After obtaining the trajectories of each day, we perform cross day matching with information gain as described in the baseline.

2.5.2.5.2 Results

We evaluate the 128-dimensional model on the same test set used in the baseline (Section 2.4.3). Here we present the result evaluated on all persistent users (users who are active every day in the test set). The set of persistent users consists of around 1300 trajectories over 15 days.



left: comparison of accuracy
right: comparison of uniqueness

Attack stage	Accuracy	Recovery Error (km)
#1	97	0.48
#2	74	4.1
#3	62	5.6

Accuracy and recovery error of our algorithm

From the above figure, our model performs marginally better in stage #1 than the baseline. And out-performs the baseline in stage #2 in terms of accuracy and average recovery error.

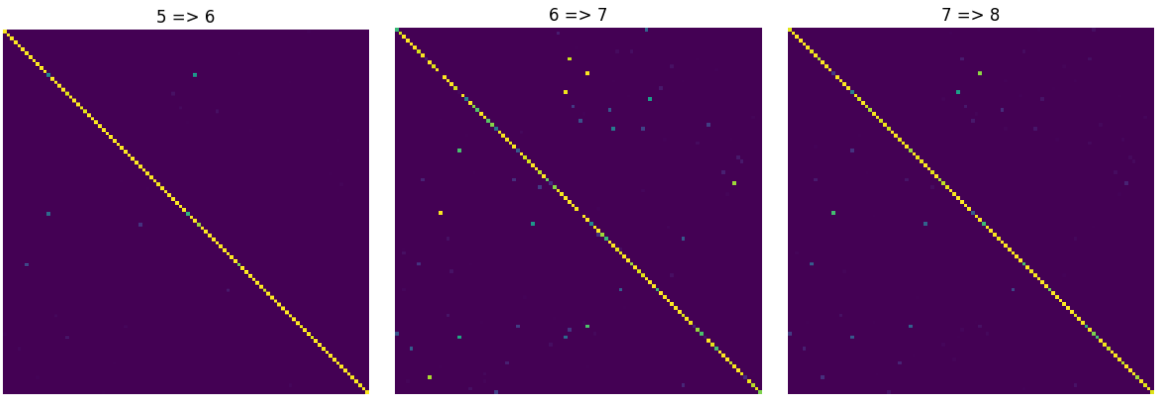
However, the trajectories recovered by our algorithm have a significantly lower uniqueness than the baseline. This indicates that the predicted trajectories are less identifiable. We hypothesize that this is due to the neural network predicting likelihoods based on common crowd behaviors in the training set.

2.5.2.6 Discussion

Since the majority of the users in the Shanghai Telecom dataset are stationary, the model can achieve a training accuracy of 96.69% simply by copying the last token in the input trajectory. Hence, it is crucial to evaluate what the model has learnt from the dataset.

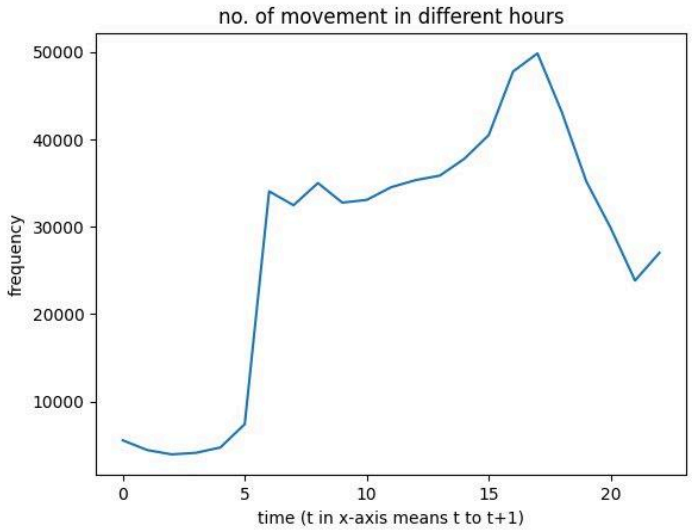
2.5.2.6.1 Embedding Quality (Temporal Properties)

We visualized the dot product similarity of base stations at different time steps and observed that the embeddings at the previous time step tends to resemble to itself at the next time step at night (before 06:00 and after 19:00).



Similarity of base different time steps, week fixed at Monday
left: 5:00 am to 6:00 am
middle: 6:00 am to 7:00 am
right: 7:00 am to 8:00 am

This indicates that the model expects more movement on day time. We plot the number of movements by timestamp below. And we observe a surge at the 6:00 am to 7:00 am interval. This coincides with our findings on the embeddings.



No. of movement at different time steps

However, we do not notice any significant differences between embeddings on weekdays and on weekends.

2.5.2.6.2 Embedding Quality (Spatial Properties)

Since most users in the training set are stationary, we are unable to conclude how well the network approximates the day time cost matrix in stage #2. But we would still like to compare our model output and the night time cost matrix used in stage #1.

We sampled a few trajectories in the validation set and compare the following:

1. top-10 tokens with highest likelihood predicted by the model
2. top-10 perturbed tokens with highest likelihood predicted by the model (perturbation done by adding normally distributed random values with variance 0.01 to the latitude and longitude)

3. 10 base stations nearest to the last token in the trajectory

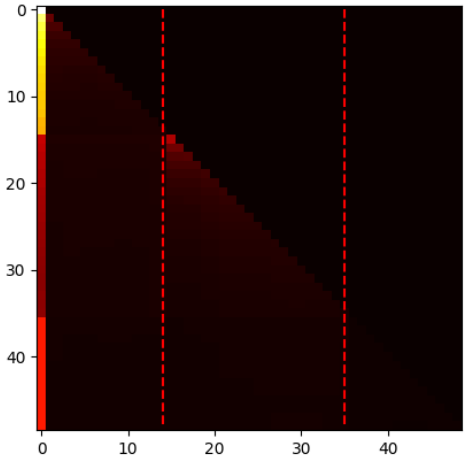
We observe that the intersection over union (IoU) similarity tends to be high (not quantified due to time constraint). And we claim that our model is a decent approximation of the night time cost matrix in stage #2.

To see how well our model can generalize to new base stations, we evaluated our model on the validation set discretized with 50 clusters.

Accuracy	Recovery Error (km)
0.9502	1.424

2.5.2.6.3 Visualization of Attention Scores

We visualized the first layer attention scores of the network in the below figure.



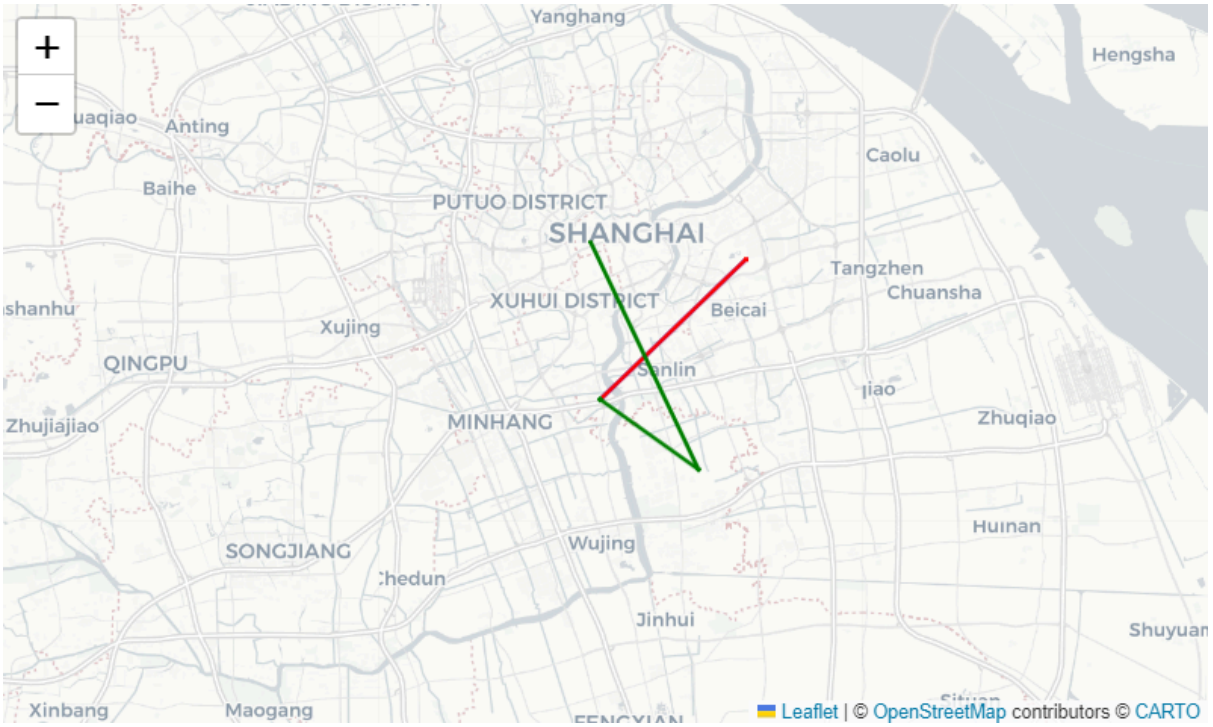
First layer attention score of a randomly sampled trajectory

We observed that a high attention score is assigned to the `[BOS]` token, this is a common pattern in transformer models reported by (Xiao et al. 2023). And we noticed another pattern in which the model tends to (not always) assign higher attention scores to the last known location of the user.

2.5.2.6.4 Generative Capability

The neural network can be used to randomly sample trajectories like a language model. We can iteratively sample the next point in time using the distribution predicted by the model.

And alternatively, we can also use the model to perform trajectory completion similar to sentence completion in language modeling. Here we iteratively append the most probable token to the trajectory until we get the entire trajectory.



Example of trajectory completion given the first point
green: generated trajectory
red: ground truth

2.5.2.7 Future Directions

2.5.2.7.1 Computational Complexity of Transformers

The computational cost of transformers is prohibitive for long sequence modeling. This may become problematic for smaller temporal resolutions, and deep learning based cross-day trajectory recovery.

Method	Time complexity
Baseline	$O(nL)$
Transformer (without cache)	$O(nL^3)$
Transformer (with cache)	$O(nL^2)$

Total cost of cost matrix computation in stage #2,
where n is the no. of trajectories, L is the sequence length (48 in our case)

We think that the computational cost can be potentially reduced by using sequence modeling architectures with linear time complexity such as Mamba (Gu & Dao 2023).

2.5.2.7.2 Multi-task Learning

Our base station embedding architecture can be reformulated as multi-task learning where the model needs to make likelihood predictions given a task context (the one-hot context vector can be interpreted as a task id) (Iyer et al. 2022).

2.5.2.7.3 Robustness

Similar to masked language modeling, we can randomly mask some of the tokens in the input trajectory with a special `[MASK]` token during training. We can train the network to be more robust to errors during inference.

2.6 Additional: User Interface Design

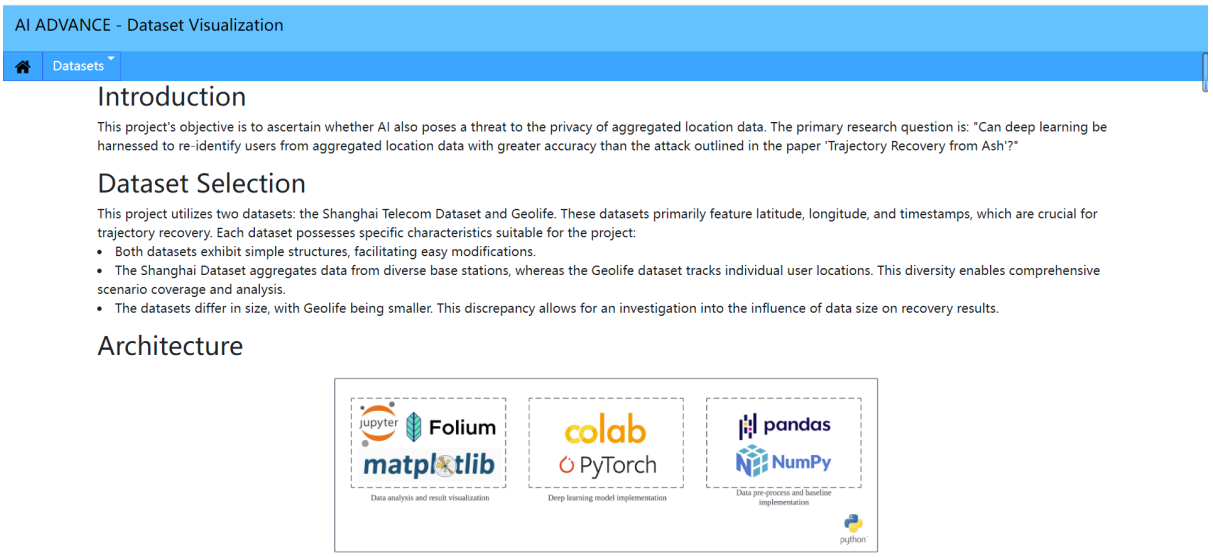
2.6.1 Webpage Interface

The interface contains mainly two sections, one header and one content section.

For the header, we designed a home icon and a dropdown menu. By clicking the home icon, the user can go back to the home page at any time. And through the dropdown menu, the user can retrieve the page containing all the information obtained from both data exploration and implemented outcomes about two datasets we have used. The function is achieved with the onClick functions.

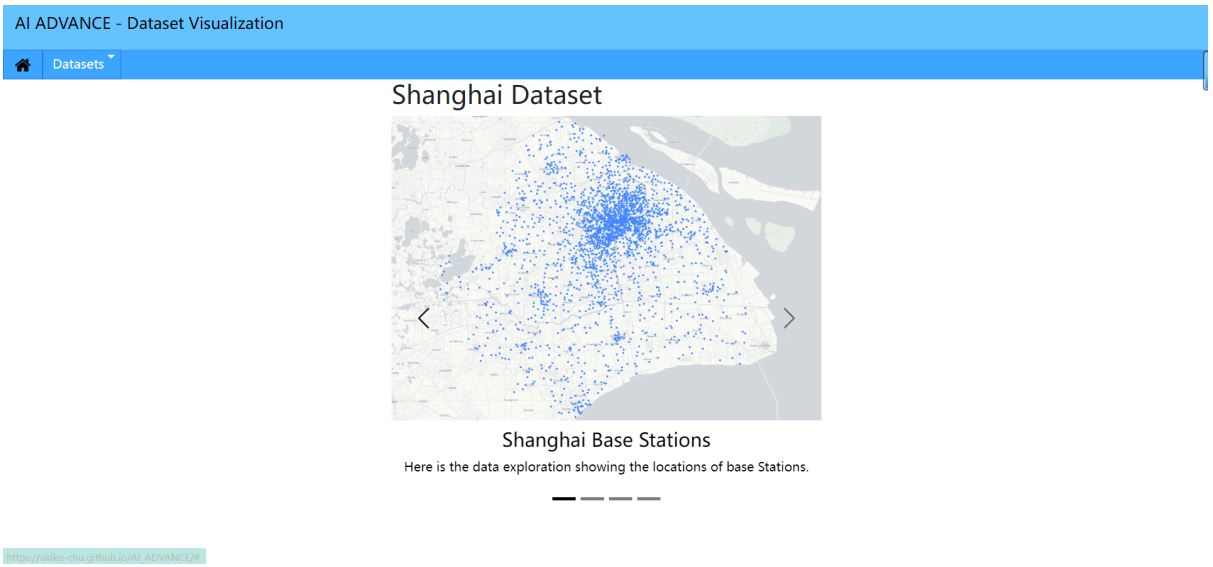
The screenshot for Header, including a Home icon and a navigator

For the home page, it is a brief introduction for our goals, our dataset usage and also our software architecture.



Home page content overview

For all the result visualization, we used a Carousel element from React-Bootstrap, which demonstrates our screenshots cyclically. For each image shown, we also have a title and a short sentence to explain the image with more detail.



Visualization page content overview

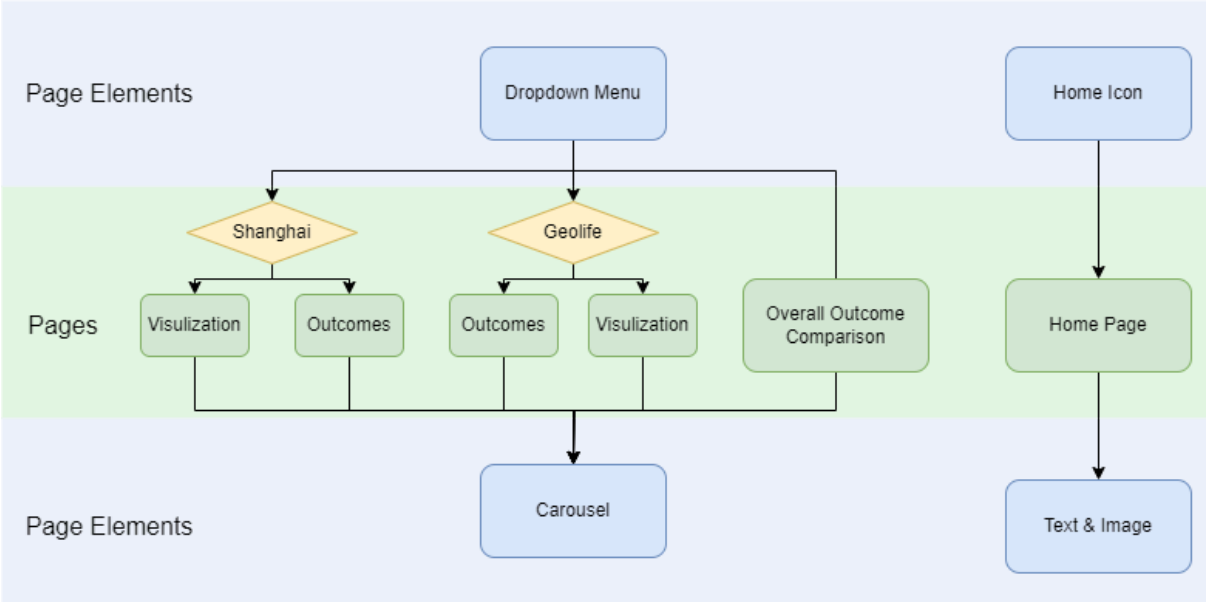
The page is constructed with the predefined method “create-react-app”, which includes the template of a webpage. The basic elements we have used are from react-bootstrap, including a Dropdown Menu and a Carousel. To reduce the code duplication and maintain consistency, the styles are defined as a CSS file and imported to each Carousel. And for each content, the only difference is the image and explanations.

```
function App() {
  const [currentPage, setCurrentPage] = useState('home');

  const handlePageChange = (page) => {
    setCurrentPage(page);
  };

  const renderPage = () => {
    switch (currentPage) {
      case 'shanghai':
        return <Shanghai />;
      case 'geo':
        return <Geolife />;
      case 'geoout':
        return <GeolifeOutcome />;
      case 'shanghaiout':
        return <ShanghaiOutcome />;
      case 'outcome':
        return <Outcome />;
      default:
        return <Home />;;
    }
  };

  return (
    <div className="App">
      <Header onPageChange={handlePageChange} />
      <div className="container">
        {renderPage(currentPage)}
      </div>
    </div>
  );
}

export default App;
```

App code - Main structure review

We have also deployed this webpage to github, so the user can view this page at any time. The link is https://akiko-chu.github.io/AI_ADVANCE/.

2.6.2 Page Flow Diagram



2.7 Epic3: Project Reflection and Future Insights

Baseline Accuracy	• Achieved lower accuracy with smaller datasets compared to the original study • Indicating limitations in the baseline method when applied to less dense data
Deep Learning Superiority	• Deep learning models demonstrated higher accuracy than the traditional algorithm used in the referenced paper • Highlighting the potential of advanced AI techniques in enhancing data analysis.
LSTM Limitations	• Direct application of LSTM models in our datasets showed significant limitations, showing its limitations with sparse and small datasets
Transformer Model Performance	• Custom-designed Transformer

	models outperformed others, offering promising results in trajectory tracking but also exposing the ongoing risks of data leakage in privacy protection

Throughout the project, we encountered a range of technical challenges, especially related to data density and model efficacy. The baseline method, when applied to smaller, less dense datasets, achieved lower accuracy than anticipated. This outcome highlighted inherent limitations in the traditional methods when dealing with sparse data.

Our exploration into deep learning models unveiled their superiority over traditional algorithms. The higher accuracy rates demonstrated by these models underscore the vast potential of advanced AI techniques to enhance data analysis capabilities significantly.

The direct application of LSTM models on our specific datasets revealed significant limitations, emphasizing the challenge of using this approach with sparse and small datasets.

On the other hand, despite these challenges, our custom-designed Transformer models showed exceptional promise. They not only outperformed other models in trajectory tracking but also brought to light the ongoing risks of data leakage in privacy protection. This dual outcome serves as a vital reminder of the complexities involved in designing data protection mechanisms in the age of AI.

Looking forward, the insights gained from this project lay a robust foundation for future studies. The limitations and successes documented here will guide the next steps in enhancing model accuracy and data handling practices.

Potential areas for future exploration include:

- Enhancing Data Density: Investigating methods to enrich sparse datasets without compromising privacy.
- Model Optimization: Continuing to refine deep learning models, particularly focusing on adapting LSTM frameworks to better handle the peculiarities of smaller datasets.
- Security Measures: Developing more robust data protection measures to safeguard against the sophisticated capabilities of AI-driven attacks.

3 Technologies Descriptions

3.1 Third-party functionalities

3.1.1 React

React is one of the most popular frameworks in the frontend. As the uses of JavaScript have increased in recent years, we now have multiple options available in the market like Vue. The

reason why we choose to use React is that it is a simple and lightweight library, and easy to use. It is a reasonable option for the beginner level front-end developer.

For React:

React. (n.d.). React - A JavaScript library for building user interfaces. Retrieved from https://reactjs.org/

3.1.2 React Bootstrap (Wernke, & Dunkel)

Bootstrap is a powerful toolkit when designing web user interfaces -- it is a collection of HTML, CSS, and JavaScript tools for creating and building web pages and web applications. Bootstrap offers consistent design by using highly reusable components. As we do not have enough frontend development knowledge, this framework helps our group to design clean web page appearances, add smooth animations, and provide useful functionalities. With some works of our own CSS

3.1.3 Other third party dependencies

All third party dependencies used in the trajectory recovery study are listed in requirements.txt on Github.

3.2 Challenges and Difficulties

In the course of our project, we encountered a series of challenges that tested our technical acumen and collaborative abilities. Each challenge demanded tailored solutions and strategic adjustments, which not only helped us overcome immediate obstacles but also enhanced our overall project execution. Below, we detail some of these challenges and the measures we took to address them.

3.2.1 Dataset Quality Issues

Problem: Our project faced significant data quality issues with the datasets we chose. The Shanghai dataset was notably sparse, making it difficult to conduct accurate analysis, while the GeoLife dataset was small, limiting the scope of our study.

Solution: To mitigate these issues, we implemented a strategy of grouping base stations in different cluster sizes in the Shanghai dataset, which helped enhance the density of data points and thus the reliability of our analysis. For the GeoLife dataset, we applied a filtering criterion that excluded data segments involving fewer than ten users. This approach helped focus our analysis on more substantial and informative data sets, thereby improving the quality of our insights.

3.2.2 Low Initial Accuracy in Baseline Model

Problem: The baseline model initially displayed disappointingly low accuracy. Attempts to improve this by swapping the information gain metric for accuracy did not yield the expected

improvements. Furthermore, the original selection of data for our Shanghai baseline involved datasets that did not closely reflect realistic user behavior, further complicating our analysis.

Solution: In response, we rewrote the baseline function to better accommodate the characteristics of our data. We also refined our data selection criteria in Shanghai to include data points at the intersection of consistent user numbers and user numbers per base station, focusing on datasets with an optimal number of 50 bases and selecting data from time periods with the highest common user count. These changes were instrumental in enhancing the model's accuracy and relevance.

3.2.3 Team Communication and Coordination

Problem: Early in the project, our team struggled with timely communication, which led to some tasks falling behind schedule and less than optimal project management. This lack of synchronization threatened to delay the entire project.

Solution: We addressed this challenge by collectively improving our communication and workflow. The team committed to regular updates and more efficient management practices, ensuring that everyone was aligned and fully informed of project progress and changes. This proactive approach not only helped us catch up and meet our original project milestones but also allowed us to expand our scope to incorporate deep learning innovations. Real-time adjustments to the project process were made as new challenges emerged, which kept the project dynamic and on track.

3.2.4 Computational bottleneck

Problem: We encountered a computational bottleneck at computing information gain at the stage #3 of the algorithm. This computation is not trivial to be vectorized unlike accuracy and recovery error. Currently it takes about 2 hours to evaluate end-to-end trajectory recovery performance.

Solution: Not implemented due to time constraints. But we propose that it can be optimized through the use of multi-threading or vectorization with sparse tensors.

4 References

Buchholz, E, Abuadbba, A, Wang, S, Nepal, S & Kanhere, SS 2022, 'Reconstruction Attack on Differential Private Trajectory Protection Mechanisms', DOI: 10.48550/arxiv.2210.09375.

Chen, G, Viana, AC, Fiore, M & Sarraute, C 2019, 'Complete trajectory reconstruction from sparse mobile phone data', *EPJ data science*, vol. 8, no. 1, pp. 1–24, DOI: 10.1140/epjds/s13688-019-0206-8.

Gu, A & Dao, T 2023, 'Mamba: Linear-Time Sequence Modeling with Selective State Spaces', DOI: 10.48550/arxiv.2312.00752

Guo, Y, Wang, S, Zhou, A, Xu, J, Yuan, J & Hsu, C 2020, 'User allocation‐aware edge cloud placement in mobile edge computing', *Software, practice & experience*, vol. 50, no. 5, pp. 489–502, DOI: 10.1002/spe.2685.

Iyer, A, Grewal, K, Velu, A, Souza, L, Forest, J & Ahmad, S 2022, 'Avoiding Catastrophe: Active Dendrites Enable Multi-Task Learning in Dynamic Environments', *Frontiers in neurorobotics*, vol. 16, DOI: 10.3389/fnbot.2022.846219

Li, Y, Zhou, A, Ma, X & Wang, S 2022, 'Profit-Aware Edge Server Placement', *IEEE internet of things journal*, vol. 9, no. 1, pp. 55–67, DOI: 10.1109/JIOT.2021.3082898.

Radford, A, Kim, JW, Hallacy, C, Ramesh, A, Goh, G, Agarwal, S, Sastry, G, Askell, A, Mishkin, P, Clark, J, Krueger, G & Sutskever, I 2021, 'Learning Transferable Visual Models From Natural Language Supervision', DOI: 10.48550/arxiv.2103.00020.

Sak, H., Senior, A., & Beaufays, F 2014, ' Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition', DOI: 10.48550/arxiv.1402.1128

Van Houdt, G., Mosquera, C., & Nápoles, G 2020, 'A review on the long short-term memory model', *Artificial Intelligence Review*, vol. 553, no. 8, pp. 5929-5955, DOI: https://doi.org/10.1007/s10462-020-09838-1

Vaswani, A, Shazeer, N, Parmar, N, Uszkoreit, J, Jones, L, Gomez, AN, Kaiser, L & Polosukhin, I 2017, 'Attention Is All You Need', DOI: 10.48550/arxiv.1706.03762.

Wang, S, Guo, Y, Zhang, N, Yang, P, Zhou, A & Shen, X 2021, 'Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach', *IEEE transactions on mobile computing*, vol. 20, no. 3, pp. 939-951, DOI: 10.1109/TMC.2019.2957804.

Wernke, B., & Dunkel, P. (n.d.). 'React-Bootstrap'. *GitHub*. Retrieved from https://github.com/react-bootstrap/react-bootstrap

Xiao, G, Tian, Y, Chen, B, Han, S & Lewis, M 2024, 'Efficient Streaming Language Models with Attention Sinks', DOI: 10.48550/arxiv.2309.17453

Xu, F, Tu, Z, Li, Y, Zhang, PY, Fu, X & Jin, D 2017, 'Trajectory Recovery From Ash: User Privacy Is NOT Preserved in Aggregated Mobility Data', DOI: 10.48550/arxiv.1702.06270