Technical Report

Salsabil Arabi
PI: Professor Tanzima Hashem

Bangladesh University of Engineering and Technology— December 20, 2020

Motivation

Digital contact tracing has a great role to play in learning epidemiology. With that motivation, I decided to work on predicting a person's future meetup information using historical check-in data. So whenever a person is infected, we can infer possible suspects who might come within his contact within a certain period. Many research have been done for next location predicting using trajectory data. Predicting next social event is also quite explored. However, finding the next friends gathering location and time of an specific person is still quite unexplored.

1 Problem definition

Given the user-specific check-in data, C = <c1,c2,c3...> of a person p and the friendship connection f = <f1,f2,f3...> of p, this work is intended to predict the next location, time and companion where p is going to meet his/her friends.

2 Dataset

To validate the proposed approach, I am using Foursquare global scale check-in dataset with user social networks. Each entry of the check-in data file contains the following info: User ID, Venue ID, UTC time, Timezone offset. The friendship network file contains the friendship information of the users. Each entry has two user id columns indicating the friendship between the two users.

3 Methodology

We can devide the problem into two parts. First, we need to find the friend co-location/meetup events of each user from random check-in data. Second, we need to perform predictive analysis. The events of step one will be the input of the model. I will discuss each step in the subsequent sections.

3.1 Finding friend meetup events

To find the friend meetup events, we need to run queries on three different files. To cope with the large size of check-in data file and varied sparsity of locations, I decided to divide the dataset into parts depending on country. To do so, I performed a merge on check-in event file and point of interest data file to add the corresponding country of each check-in location and divided the check-in data file of each country separately. Then, I performed inner join on two similar check-in data file to find pair of users who were in the same POI. I processed temporal query to extract the event when the both users were at the same POI within a close time. Finally, I merged the friendship network to extract the entries where two co-located users were friends. For this project, I have considered these entries as friend meetup events. All the queries were processed using AWS S3 and Athena.

3.2 Predictive analysis

In this step, I am basically trying to predict the next meetup location. In order to finalize solution approach, I initially preprocessed the dataset. I conducted exploratory analysis on some samples to see if there lies any pattern in each users meetup pattern.

3.2.1 Exploratpry Analysis

I conducted exploratory analysis on some samples to see if there lies any pattern in each users meetup pattern. Also, it will help me to extract features and reduce dimensions.
Initially I cleaned the dataset and checked if there lies any null or missing value. There were just a few missing value issues in the dataset. I also checked the mean, median, standard deviation and other percentile values of the dataset to know if there is any extreme values or outliers in the dataset. The values indicate that there are some issues in the longitude values (add image). After plotting the events of some

	userid	time_in_minute	userid2	time_in_minute2	latitude	longitude
count	3.811400e+04	3.811400e+04	3.811400e+04	3.811400e+04	38114.000000	38114.000000
mean	5.294193e+05	1.057909e+09	5.294193e+05	1.057909e+09	37.051019	-93.747555
std	5.723931e+05	2.127748e+05	5.723931e+05	2.127748e+05	5.992961	21.013805
min	1.900000e+01	1.057685e+09	1.900000e+01	1.057685e+09	18.412803	-159.523949
25%	1.160190e+05	1.057740e+09	1.160190e+05	1.057740e+09	33.789135	-104.890294
50%	2.601610e+05	1.057816e+09	2.601610e+05	1.057816e+09	38.916740	-87.621754
75%	8.148350e+05	1.058056e+09	8.148350e+05	1.058056e+09	41.193846	-78.760519
max	2.181131e+06	1.058643e+09	2.181131e+06	1.058643e+09	63.738315	-66.253910

Figure 1: Statistics of the dataset

samples, I observed that the meetup pattern of an individual has some recurrency (add image). So, I decided to use Long Short Term Memory (LSTM) Recurrent Neural Network model for the prediction of location. In order to observe the correlation among variables, I used pandas .corr() function and ob-

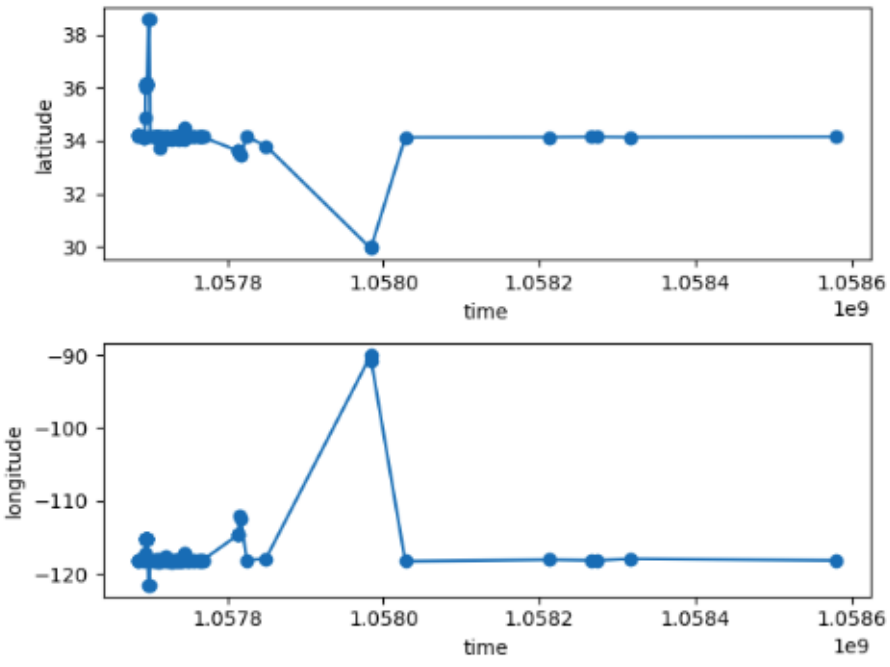

Figure 2: meetup events of user with id 470

served the correlation matrix using a heatmap in seaborn library(add image). The darker shades represent positive correlation and the lighter shades represent negative correlation. As we are not applying linear regression model, we do not need to bother much regarding correlation among features. However, while extracting features, we can choose to remove correlated variables to reduce dimension.

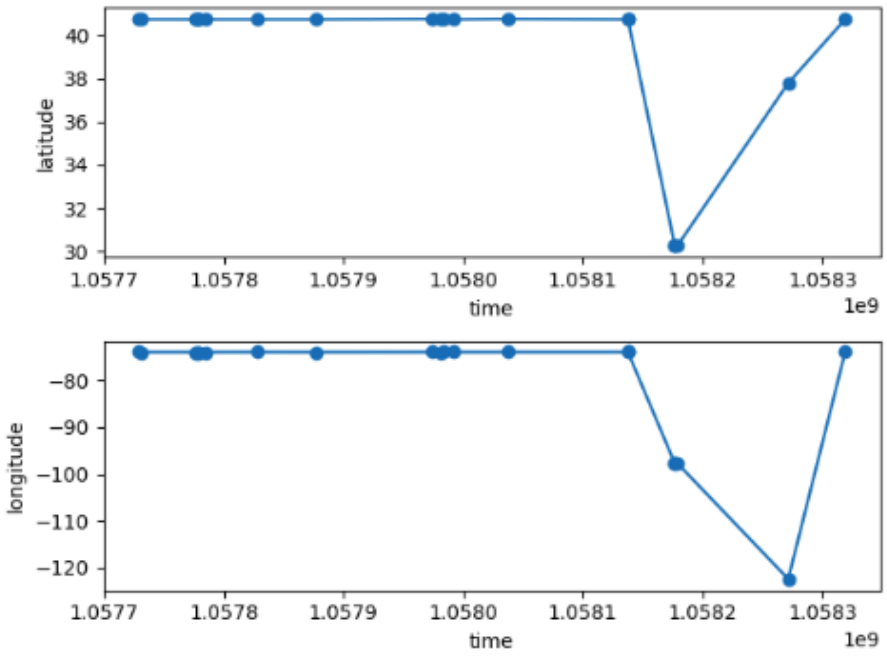

Figure 3: meetup events of user with id 580

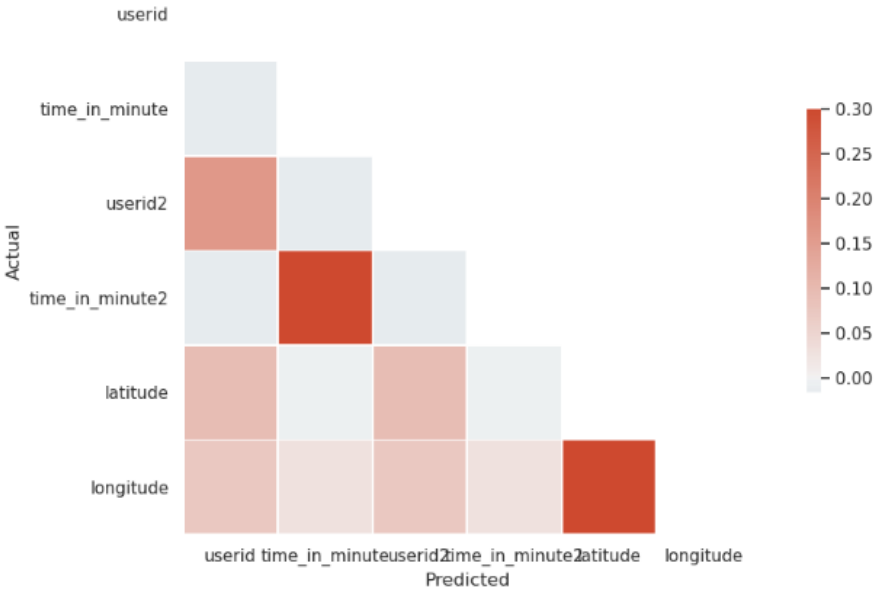

Figure 4: correlation matrix among features

3.2.2 Clustering

I used meanshift clustering algorithm of scikit-learn library in order to divide the input check-in locations into clusters(add image). In the output layer, I will get the probabilities of being in each cluster. To get the actual longitude and latitude, I have multiplied the probabilities with the cluster centers.

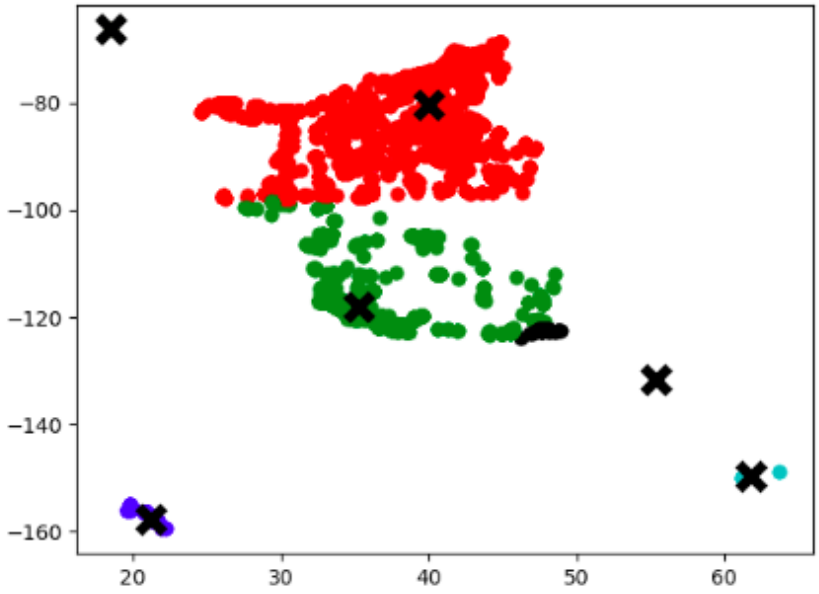

Figure 5: Clustering of US based events

3.2.3 Preprocess Dataset

Before feeding the model, I needed to process the input dataset. I converted the feature values into categorical values using labelencoder function from scikit learn library. Also, I removed some correlated features to reduce feature dimension. The final set of feature is venueid, cluster grp, day, hour, date, holiday, year, longitude, and latitude.

3.2.4 Building the model

I have used the LSTM model from keras library to perform predictive analysis. The model is comprosed of an input layer, a hidden layer of 100 neurons and RELU activation function, and an output layer. The output layer has one unit for each cluster group. I have used the softmax activation function in this layer. Also, I have used categorical crossentropy as the loss function. We will multiply the probability outcomes of the model with the cluster centers to get the predicted longitude and latitude.

3.2.5 Training the model

I trained the model over 10 epochs. From a total of 2780 test data, the model accurately identified the cluster groups of 2123 data. I still need to perform other optimization to get better accuracy of clusters and locations. (image)

```
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  40.096253000000004 lat  -88.235879
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  39.914274 lat  -86.104488
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  32.498435 lat  -92.060069
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  33.785740000000004 lat  -84.384351
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  36.088964000000004 lat  -115.177982
tf.Tensor([[ 38.760616 -92.841156]], shape=(1, 2), dtype=float32) long  42.35947 lat  -71.053726
tf.Tensor([[ 38.761288 -92.80704 ]], shape=(1, 2), dtype=float32) long  45.141501 lat  -93.23420300000001
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  32.507384 lat  -84.963313
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  42.351544 lat  -71.064875
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  45.534715999999996 lat  -122.65554499999999
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  39.958562 lat  -75.136658
tf.Tensor([[ 38.761288 -92.80686 ]], shape=(1, 2), dtype=float32) long  35.227677 lat  -80.839084
total  2780  correct  2123
```

Figure 6: Snippet of test data prediction