Point-of-Interest Clustering Based on Trajectory Mining

Saim Mehmood

Electrical Engineering and Computer Science
York University

saim@eecs.yorku.ca

Abstract - Use of location finding apps and prevalence of location based services have made it easier to record movement/trajectories of objects such as humans, cars, aeroplanes & animals etc. These trajectories offer a possibility to understand object behavior in terms of interaction with Point-of-Interest (POI). The idea behind this paper is to visualize POI-POI correlation using node2vec and t-SNE to find similarities between different POI based on the movement trajectory of objects. This in turn can be used for POI recommendation.

Keywords - POI recommendation, Trajectory Data Mining, Visualization, Graph Embedding.

I. Introduction

With the proliferation of smartphone devices and social networks, it is easier to track user location and movement behavior. Specifically, location finding apps such as Google Maps keeps track of the user travelling history. Our goal is to use trajectories of objects and POIs that are near to their movement path, in order to understand interactions between the both. Among our goals is to cluster points-of-interest, based on the number of times they get visited sequentially by the same object.

A common factor in the previous work is related to utilizing only specific datasets (such as Yelp, Foursquare, Facebook Places, Gowalla and Loopt etc) to visualize POI-POI graph embeddings [6]. This lack the generalization across POI clustering because of the cold start and data sparsity problems. In this work, we are utilizing Google Places API to retrieve datasets of places (POIs) which covers dense cities and suburban areas.

The idea is to get all POIs around a specific area and retrieve information related to every place of interest. Then relate these POIs with movement trajectories of objects based on a threshold value. Using state-of-the-art graph/network based embedding approaches such as node2vec [1], we will be generating embeddings of POIs. To visualize embeddings a popular method called t-SNE proposed by van der Maaten and Hinton in 2008 [2] will be implemented.

II. Related Work

Our work falls under the categories of Trajectory Data Mining, POI recommendation and Graph Embedding.

POI recommendation: Researchers in the area of POI recommendation have looked at various aspects of human mobility. Bin Liu et al. performed an integrated analysis of the joint effect of *user preferences, geographical influences* and *user mobility* to find geographical preferences for point-of-interest recommendation [3]. Tobler's first law of geography is also an interesting concept to consider while thinking about POI correlation i.e., "Everything is related to everything else, but near things are more related than distant things" [4].

Trajectory Data Mining: The advances in location-acquisition and mobile computing techniques have generated massive spatial trajectory data, which represents the mobility of a diversity of moving objects, such as people, vehicles, and animals. In this survey paper the author have explored the connections, correlations, and differences among trajectory pattern mining, trajectory data preprocessing, outlier detection and trajectory classification techniques [7].

Graph Embedding: In *node2vec* [1], they propose an algorithmic framework for learning continuous feature representations for nodes in networks. There key contribution is in defining a flexible notion of a node's network neighborhood. By choosing an appropriate notion using random walks, *node2vec* can learn representations that organize nodes based on their network roles and communities. Van der Maaten et al. introduced a technique called *t-SNE* that helps in visualizing high-dimensional data in a two or three-dimensional space [2].

III. definitions and preliminaries

A. Problem Description

POI: A POI is defined as a uniquely identified place such as a restaurant or a cinema etc. In our work we are retrieving

POIs using Google Places API [5] inside a specified geographical area. We use P_i to denote point-of-interest.

$$POI = \{P_1, P_2, \ldots P_i, \ldots P_n\}$$

where P_1, P_2 are individual points of interest.

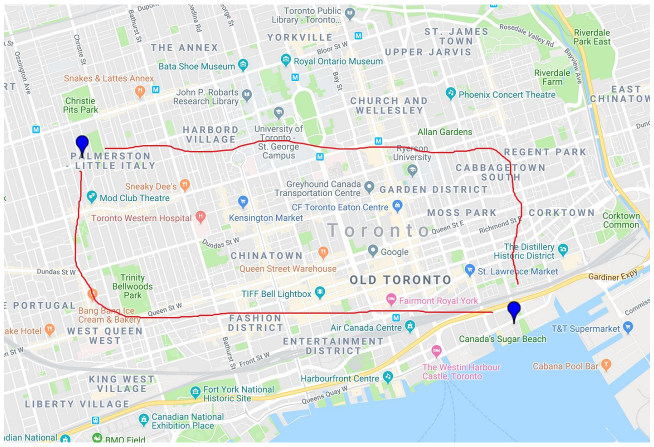

Fig 1: Retrieving unique POIs inside the rectangle

Querying Process: To query POIs from the Google Places API, certain parameters are required to initiate a search request. We have written code [9] around the API and are using *location* and *radius* parameters to retrieve POIs data [8].

Parameters: location - the latitude/longitude around which to retrieve place information. This must be specified as (latitude, longitude).
radius - defines the distance (in meters) within which to return place results. The maximum allowed radius is 50,000 meters. Results inside of this region will be ranked higher than results outside of the search circle; however, prominent results from outside of the search radius may be included.

Step Value: This value identifies the distance between POIs retrieval based on the Places API query. It means how far away from the previous point on the map we are querying from. For Example, by increasing the step value from 0.01 to 0.001 (meters), we get different results of fetched values (shown in Fig 2 and 3).

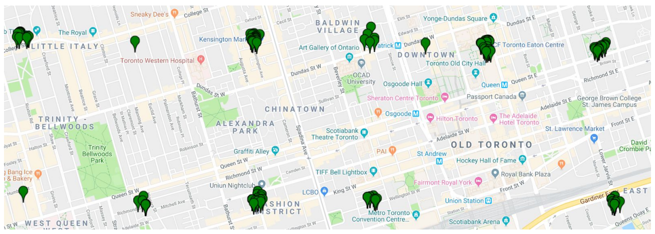

Fig 2: Green points representing fetched POIs based on the step value 0.01

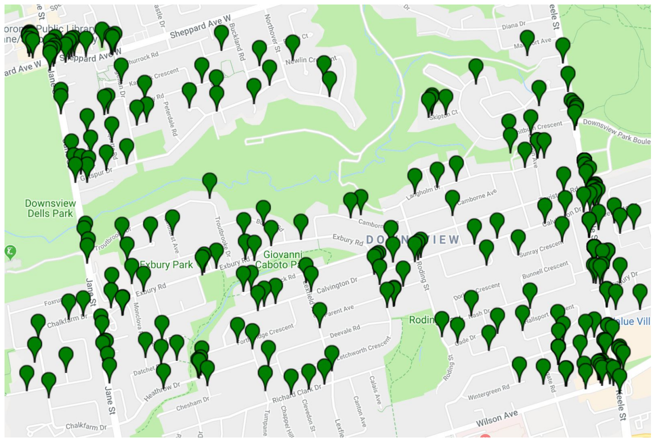

Fig 3: Green points representing fetched POIs based on the step value 0.001

Cold start and data sparsity problems: If a user has no trajectory history, it is difficult to establish predictors regarding his future location, which is known as the cold start problem. If a user has only a few visiting locations, it is also difficult to establish predictors of his future location, which is a data sparsity problem. Cold start and data sparsity problems are prevalent in prediction and recommendation applications, especially those using actively recorded trajectory data [11].

Trajectory: Trajectory is the path that an object in motion follows through space as a function of time. We define it as a route traveled from a starting point to an ending point using variety of travelling modes e.g., bicycling, walking and transit etc. We use T_i to denote a trajectory.

$$T_i = \{(x_0, y_0, t), (x_1, y_1, t), \ldots (x_i, y_i, t)\}$$

where (x_0, y_0, t) and (x_i, y_i, t) are the set of points inside Trajectory T_i at time $t = 0$ *and* $t = i$ etc.

Querying Process: Similarly, we are fetching trajectories from Google Direction API by specifying different travelling modes i.e., driving, walking, transit and bicycling etc.

Plotting: To plot POIs and Trajectories we are using *gmplot* library to visualize the functionality of different parameters and the way POIs are fetched from API.

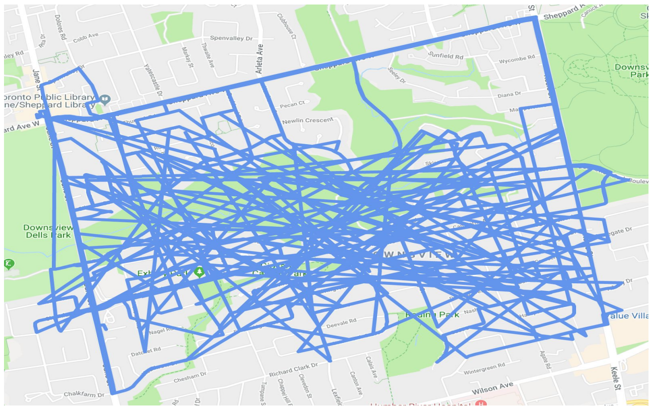

Fig 4: Retrieving and Plotting 100 trajectories

B. Formal Definition

Given a set of N points-of-interest P_n, and N number of trajectories T_n around a specific area, our goal is to associate POIs with each trajectory T based on a threshold ϑ value. Whereas, ϑ is the distance between any single trajectory point (x, y, t) and point of interest P_i.

$$P^T \subseteq POI = \{P_1, P_2, \dots P_i, \dots P_n\}$$

where P^T represents POIs that belongs to trajectory T.

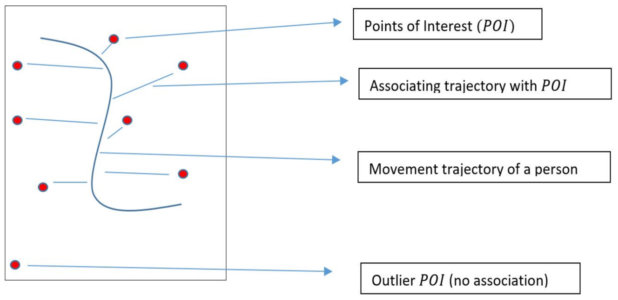
Points of Interest (*POI*)
Associating trajectory with *POI*
Movement trajectory of a person
Outlier *POI* (no association)

Fig 5: Trajectory path inside a specified area

C. Graph Construction

We are getting point-of-interest (POI) and random trajectories using Google Places and Google Direction APIs respectively. Considering these random trajectories as movement of people and associating them with POIs based on a threshold ϑ value, we will construct a graph. Nodes are the set of places visited by user and edges will denote connections between them.

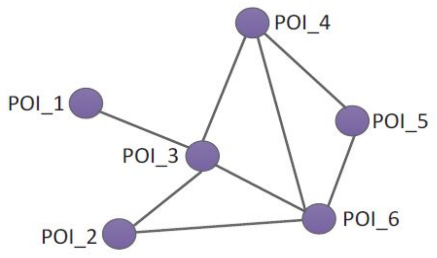

Fig 6: POI-POI Graph

The POI-POI graph (Fig 5) captures the sequential patterns of POIs visited by user. For example, a person having a dinner at a restaurant on weekend could be visiting a bar afterwards. Based on this movement, we can generate an edge between the bar and restaurant. Intuitively, if v_i and v_j are often visited sequentially, their correlation will be higher i.e., if a user has visited v_i, he has a high probability to visit v_j next.

Grid Construction: Instead of using Euclidean distance between POI and trajectory for association. We are dividing the area into small grids. The advantage comes in the form of avoiding the cost of calculating distances between all POIs and trajectories.

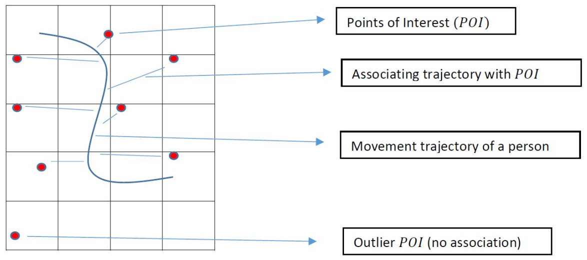
Points of Interest (*POI*)
Associating trajectory with *POI*
Movement trajectory of a person
Outlier *POI* (no association)

Fig 7: Trajectory path inside a specified area (divided into grids)

Trajectory Formations: A trajectory inside an area divided into grids can be written as:

$$T_i^C = \{C_{11}, C_{21}, \dots, C_{ij}\}$$

where C_{ij} denotes the grid cell containing the portion of trajectory. By associating all the POIs inside a single cell with trajectory T_i, it can be written as:

$$T_i^{POI} = \{POI_{C11}, POI_{C21}, \dots, POI_{Cij}\}$$

where POI_{Cij} denotes the set of POIs inside the grid cell C_{ij} . The trajectory information can be further transformed into a trajectory containing POIs based on random walks.

$$T_i^{RW} = \{POI_1, POI_2, \dots, POI_i\}$$

By using random walks, we are imitating the movement of user from one place to another. This movement in turn can be visualized using t-SNE to find correlation among point-of-interest. The places that will appear together the most will have high correlation.

IV. Experimental Evaluation

In order to evaluate this work, we will compare it with other benchmarking POI recommendation frameworks.

V. Conclusion

This paper talks about finding correlation between *Point-of-Interest* (POI) based on the movement trajectory of objects and their interaction with them. This work introduces novelty in the form of POIs & Trajectories retrieval by using Google API's. This way we are avoiding the data sparsity and cold start problems mentioned in the previous work [6, 11, 12].

Dividing geographical area into grids adds innovation to the existing work by avoiding the cost of calculating Euclidean distance between every POI and trajectory. Instead we only need to calculate distance between portion of a trajectory and POIs residing in the specific cell of grid.

As a future work, we will look into sequential models in order to better understand and find correlations among POIs.

V. Reproducibility

The source code and some execution examples are publicly available to encourage reproducibility of results. They can be accessed at the following website:
https://github.com/saimmehmood/POI_Clustering

VII. References

[1] *Aditya Grover, Jure Leskovec, "node2vec: Scalable Feature Learning for Networks," In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.*

[2] *Maaten, L.v.d. and Hinton, G., "Visualizing data using t-SNE", Journal of Machine Learning Research, Vol 9 (Nov), pp. 2579 - 2605, 2018.*

[3] *Bin Liu, Yanjie Fu, Zijun Yao, Hui Xiong, "Learning geographical preferences for point-of-interest recommendation", KDD'13 Pages 1043 - 1051.*

[4] *W. Tobler, "A computer movie simulating urban growth in the detroit region", Economic Geography, 46 (2): 234 - 240, 1970.*

[5] *https://developers.google.com/places/web-service/intro*

[6] *Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, Sen Wang, "Learning Graph-based POI Embedding for Location-based Recommendation", CIKM'16 Pages 15-24.*

[7] *Yu Zheng, "Trajectory Data Mining: An Overview", ACM Transaction on Intelligent Systems and Technology, September 2015*

[8] *https://developers.google.com/places/web-service/search*

[9] *https://github.com/saimmehmood/POI_Clustering*

[10] *https://github.com/vgm64/gmplot*

[11] *Ruizhi Wu, Guangchun Luo, Junming Shao, Ling Tian, and Chengzong Peng, "Location Prediction on Trajectory Data: A Review" Big Data Mining and Analytics pp108-127*

[12] *Huayu Li, Yong Ge, Richang Hong and Hengshu Zhu, "Point-of-Interest Recommendations: Learning Potential Check-ins from Friends" KDD'16 ACM SIGKDD Pages 975-984*