Spatio-Temporal Site Recommendation

Blinded for Double-Blind review

***Abstract*—Recommendation systems have become extremely common in recent years, and are utilized in a variety of areas to predict the "rating" or "preference" that a user would give to a point of interest (PoI), such as a restaurant, a hotel, or a bar. Such systems typically produce a list of recommendations by considering previous ratings of the user, as well as ratings of other users. Not every person rates every point of interest they visit. In this work, we want to explore the use of spatio-temporal data to improve recommendation systems: We postulate that spatio-temporal user data may indicate the liking or disliking of a point of interest. Clearly, if a user frequently visits the same PoI, stays at the PoI for long times, and is willing to travel a long distance to visit a PoI, that might indicate that user likes that PoI. Thus, we propose to extract user-PoI relation features from spatio-temporal trajectory data only. Using these features, we use out-of-the-box data mining and machine learning solutions, to estimate the popularity of a PoI. Our experimental evaluation shows, that the features extracted from spatio-temporal data able to accurately predict the popularity of a PoI, using ground-truth data from FourSquare as a baseline.**

I. Introduction

Modern technology to capture geo-spatial information produces a huge flood of individual trajectory data, coupled with a new user mentality of utilizing this technology to voluntarily share information. McKinsey Global Institute [11] projects a "$600 billion potential annual consumer surplus from using personal location data globally". Towards this goal of making location data actionable, we propose to mine potential user ratings for location sites (e.g., restaurants, shops) from spatial-temporal data. User ratings are critical in many applications, especially in recommendation system. Techniques such as widely used collaborative filtering use known user rating information to estimate the interests of a user. However, in practice, the user-site rating matrix is usually highly sparse, meaning that there are not enough user rating information to perform such tasks. This is know as the cold-start problem. There are many reasons, for example, many users do not want to spend time to rate locations they visited, or locations such as shops do not have an efficient way to ask users for feedback.

To solve this problem, we propose to obtain implicit user rating rather than explicit user ratings. Our approach uses trajectory data to find users that have likely visited a site. Since this data does not explicitly tell us whether the user likes that site, we propose to mine features from the users trajectory, which we intuitively expect to implicitly describe whether the user likes that site. The major features that we propose to obtain from trajectory data include:

- **The frequency of visits of a user.** If a user visits a site only once in their life, chances are that the user did not like the site enough to return. If the site if frequented often by the user, he seems to like it.
- **The length of stay of a user.** If a user stays at a restaurant or a hotel for a long time, that indicates that he likes the site.
- **The distance to their home base.** If a user is willing to take a long journey to reach a site, thus bypassing other, similar sites, that indicates that the user is subject to a strong attraction from that site, indicating that the user likes that site.

There are several advantages of using location data for location rating: (i) A potential solution to the cold-start problem, (ii) no user effort to capture their recommendation such as filling in rating forms, and (iii) it is based on user's behavior, thus more objective and prone to alteration by fake-user-ratings and spam/bot-user-ratings.

Our approach becomes viable, due to the abundance of large open-sources collections of voluntarily contributed trajectory data, including the following data sources:

- **Location-Based Social Networks (LBSNs)** allows user to "Check-in" into a physical site such as a hotel, a restaurant or a metro station. Fairly large LBSN datasets, made anonymous, are available publicly. For instance, the FourSquare dataset used in [19] contains more than 30 million checkins and is available publicly. Such data explicitly includes the sites that a user has visited.
- **Geocoded Social Media Data:** is obtainable from public streaming APIs including for Twitter, Instagram, and Flickr. These data sources provide low-frequency trajectory data. Yet, microblogs and images are often published in sites of interest to a user, thus giving implicit information about the users site preferences.
- As part of the effort to create **Open-Street-Map (OSM)** [12], [2] road network data, users have been uploading GPS traces of their routes to the OSM site. While these routes are typically used to digitize the road network, the majority of routes is uploaded by pedestrians, and can be used as an indicator of sites that the corresponding user frequents. These trajectories are publicly available through the OSM API.
- Captured by operating companies of mobile applications. For instance, the augmented reality game "Pokémon Go"

has been downloaded more than 100 million times on Android devices alone [**?**].

The goal of this work is making this plethora of trajectory data actionable to improve everyday's life, by recommending better and more individualized sites to users. To describe our approach of site-recommendation using trajectory data, the rest of this work is organized as follows. We survey the state-of-the-art on recommender systems and location-based recommendation in Section II. Then, we formally define the problem location-based site recommendation in Section III. Our solution, using deep-learning to bridge the gap from user-behaviour to user-site-recommendations is given in Section V. Our solution is evaluated in Section VII, showing that our rating prediction for a restaurant is able to closely predict authoritative ground-truth site-ratings obtained from Yelp. We conclude our work in Section VIII.

II. Related Work

In a recommender system, there is a set of users and a set of items. The goal of of the system is to recommend a user certain items that best match the user's preference. The essential research problem here is, how to predict a user's rating for items that were not rated by him. This is due to the fact that the number of users and items are usually very large, and it is impractical to ask users to rate every item. We study this problem in the context of user-site recommendations.

Two types of recommender systems have been developed in the past few decades. **Content-based recommender systems** (e.g., [8], [13]) analyze the properties of item (e.g., item descriptions) and/or user profile to identify items that are attractive to the user. The spatial-temporal features we explored is similar to this type of recommendation. Nevertheless, existing methods mostly focus on features extracted from user's interaction with the recommender system (e.g., view or purchase history, reviews, account profile). To our knowledge, this is the first work that explores user's visiting features for user rating prediction.

The other type is **Collaborative-filtering recommender systems** (e.g., [14], [10], [5])). Collaborative-filtering predicts users' preference to items based on their similarity to other users. It relies on analysis of large amount existing product rating data. However, it becomes challenging to calculate user-similarity when there is not enough such data, which is known as the cold-start problem. A widely used technique for collaborative-filtering is Matrix factorization [6]. Matrix factorization works on a user-item rating matrix. It models both users and items as vectors of latent features.

Our work is different from Location Recommendation [20], [17], which also considers locations as recommended items. However, they propose to explore geo-social activities of users to facilitate the identification of users with similar preference, which fall in the category of collaborative-filtering. In contrast, our work is to direct predict a user's preference based on his spatial-temporal features.

III. Problem Definition

In this section we formally define a user-trajectory, and define our notion of a user-site-stay, which we are going to use to extract features to estimate the affinity between a user and a site later in Section V. We first start by defining a trajectory.

Definition 1 (Spatio-Temporal Database): Let $\mathcal{U}$ denote a set of unique user identifiers, let $\mathcal{G} = [-90, 90] \times [-180, 180]$ denote the space of longitude/latitude geo-coordinates, and the $\mathcal{T}$ denote the time domain. A *spatio-temporal database* $\mathcal{ST} \subseteq \mathcal{U} \times \mathcal{G} \times \mathcal{T}$ is a collection of triples $(id \in \mathcal{U}, (lat, long) \in \mathcal{G}, t \in \mathcal{T})$. Each triple $(u, s, t) \in \mathcal{ST}$ is called an observation.

We group a spatio-temporal database into observations of the same user, denoted as user-trajectory, formally:

Definition 2 (User-Trajectory): Let $\mathcal{ST}$ be a spatio-temporal database and let $u \in \mathcal{U}$ be a user. The set

$$\mathcal{ST}(u) := \{(u', (lat, long), t) \in \mathcal{ST} | u = u'\}$$

is called the user-trajectory of user u.

In order to obtain recommendation information from a user-trajectory, we need to link the user-trajectory to sites, such as restaurants and hotels. For this purpose, we join a spatio-temporal database with a database of points of interest (such as provided by Open-Street Map) like restaurants and hotels. Next, we define our notion of a *stay*. A stay is an event of user visiting a site, enriched by the duration of the stay.

Definition 3 (Stay Trajectory): Let $\mathcal{ST}$ be a spatio-temporal database and let $\mathcal{S} \subseteq \mathcal{G}$ be a collection of $(lat, long)$ pairs of sites. A stay is a triple $(u \in U, s \in \mathcal{S}, (t_{\text{start}}, t_{\text{end}}) \in T \times T)$, indicating that user u has stayed at site s from time t_{start} to time t_{end}. We let $(\mathcal{ST} \bowtie \mathcal{S})$ denote the set of all stays mined from all trajectories $\mathcal{ST}$ using all sites in $\mathcal{S}$. The sequence of all stays a user $u \in \mathcal{U}$ is called the stay-trajectory $(\mathcal{ST} \bowtie \mathcal{S})(u)$ of u, defined as:

$$(\mathcal{ST} \bowtie \mathcal{S})(u) := \{(u, p, (t_{\text{start}}, t_{\text{end}})) \in \mathcal{S} | u = u'\}$$

Finding stay points in trajectory and PoI databases is a research topic that has raised attention in the past. In this work, we make the explicit assumption that stay points of a trajectory are already given, as we use trajectory datasets where stay points are explicitly labeled. We discuss this assumption and related work on stay-point detection in Section **??**.

Given stay-trajectories for each user, the challenge of this work is to predict the rating between users and a site. As site is PoI which can be rated by a user, such as a restaurant or a hotel. We assume that we have a recommendation database, where users can rate sites. We assume normalized rating values in the interval $[0, 1]$, where 0 corresponds to the lowest rating and 1 corresponds to the highest rating.

Definition 4 (User-Site Recommendation Database): A recommendation database $\mathcal{R}$ is a set of user ratings $\mathcal{R} \subseteq \mathcal{U} \times \mathcal{S} \times [0, 1]$.

The task of this work is to predict the triples in R. That is, the challenge is to predict the rating that a user $u \in \mathcal{U}$ will give to a site $s \in \mathcal{S}$, using a spatio-temporal given in $\mathcal{ST}$.

IV. Discussion: Stay Point Detection

The first step of our user-site-recommendation approach requires to map a raw trajectory $\mathcal{ST}(u)$ of a user u to a stay-trajectory $(\mathcal{ST} \bowtie \mathcal{S})(u)$. Such stay points can be detected by using existing work such as proposed in [9], [23], [22], [18]. All of these works use a distance threshold θ_d and defines a stay as the duration of time where the trajectory does not exceed a distance of θ_d to a PoI. In this work, we entirely circumvent the step of implicit stay point detection, by using data that has explicit stay points. Therefore, in our experimental evaluation we use Check-in data from location based social networks (LBSNs), which explicitly contain the stay points of users. Clearly, by circumventing the problem of stay point detection, we limit our experimental evaluation to LBSN Check-in data, which is relatively small (hundreds of megabytes of data), compared to large raw trajectory databases (Terrabytes of data). Yet, we postulate that our relatively small FourSquare Check-in dataset [19], allows to effectively predict the user rating of a restaurant. This hypothesis is also supported by our experiments.

While we circumvent the problem of stay point detection by using Check-in data, we still need to estimate the duration of stay, as the duration of a stay is one of the features that we will use for prediction in Section V. Thus, we interpret each Check-in c as a stay point. If the corresponding user has no other check-ins for the next six hours, we set the duration of c to *UNKNOWN*. Otherwise, the set the duration to the time in-between these two check-ins.

V. Spatio-Temporal User-Site Feature Extraction

Our goal is to extract a set of spatial-temporal features from a user's trajectory, which can be used to predict the rating of a PoI from the users who have visited the PoI. There are many features that could potentially be related to a user's feeling of a PoI. We discuss in the section a set of common features we select for the rating prediction problem, and the methods we use to extra such features. It is easy to understand why these features are selected since they are intuitive.

A. Extracting the frequency of visits of a user

Our assumption is, if a user repeatedly visits a PoI, e.g., always dine in the same restaurant, it strongly suggests that the user favours the PoI. On the other hand, if a user visits a PoI only once and never comes back, it suggests the user dislike the place. Therefore we choose the frequency of visits as a feature. Here, each stay-point is considered a visit.

In order to extra frequency information from a user's trajectories, we count for how many time does the user visit a location in a given time period. Typically, each check-in to the location is counted as one visit. Users are then assigned into several frequency groups (Table I) based on how often they visit the location. Note that if a user has never visited a PoI, he cannot rate it. Thus we do not assign any group for such users.

We note a user's behaviour may vary over time. For example, if a user likes a shopping center, he may visit the shopping center very frequently during Thanksgiving and similar holidays, while visit the same place only once in several month during the rest of the year. As a result, the timing window used to count visits can have significant impact on the user's group assignment. For the same user, if we consider only visits happened in the Thanksgiving week, the user belongs to the "At least one visit per week " group. However, if we look at the year-long visits, he may be grouped into "Less than one visit per month" since his visits are averaged over twelve months. We propose a simple way to mitigate this problem.

TABLE I: Groups based on how often user visits a location

ID	**Frequency groups**
1	At least one visit per day
2	At least one visit per week
3	At least one visit per month
4	Less than one visit per month

First, we use timing windows with different sizes simultaneously to compute the frequency of visits of a user. Specifically, given a user's trajectory data for a period of n days, for each week/2 weeks/month within these n days, we count the number of visits that falls in the same week/2 weeks/month. We then use this count to compute the user's group of that specific period. Here we use week and month as nature timing windows, but in practice the size of timing window can be arbitrary. Second, when a user's visit frequency demonstrates inconsistency in different time period with the same timing window size, we use his maximal visiting frequency, e.g., if a user visits a location 3 times a week in week 1, but 0 times in week 2, the user is then categorized as "At least one visit per week" in this case.

Additionally, for each PoI, we calculate three numerical features that also describe a user's visiting pattern to the PoI: 1) **Average duration between two consecutive visits of the user**, 2) **Minimal duration between two consecutive visits of the user**, and 3) **Maximal duration between two consecutive visits of the user**. These information are supplemental to the frequency groups.

B. Extracting the length of stays of a user

Given a trajectory of a user, the length of stays can be inferred by examining the interval between consecutive check-ins of the user. If two visits are reported within a time threshold τ, e.g., 30-minutes, it is safe to consider the two check-ins belong to the same stay. Thus, the minimal length of this stay is the time frame between the two check-ins. Similarly, an upper bound of the length of stay is the time between a check-in to the location and the closest check-in to a different location. This provides us with a coarse way to estimate the average length of stay of users to a PoI. This method is most accurate if the user's location is reported periodically with small intervals, or whenever the user changes location.

C. Extracting the travel distance of a user from its home base

If a user is willing to travel a long distance from his home/work area to a PoI, it is likely that the PoI is attractive to him. Unfortunately, measuring the travel distance from one's home/work can be challenging. This is because users' home or work address or coordinates are usually not explicitly marked in a spatial-temporal dataset due to privacy concerns.

We use distance-based location clustering (e.g., weighted k-means [4], where the weight of a location is the number of visits by the user) to estimate a user's home base, i.e., an area where he is likely to live/work. Locations a user visits frequently is likely to formulate a dense cluster in the area where he lives and works [1], [21]. If there exists a location that is isolated from any of these clusters, we deem such a location as "far from home". It requires extra effort for the user to travel this location. Willing to make such effort indicates the user may like the place. An example of home base and isolated locations is illustrated in Figure 1.

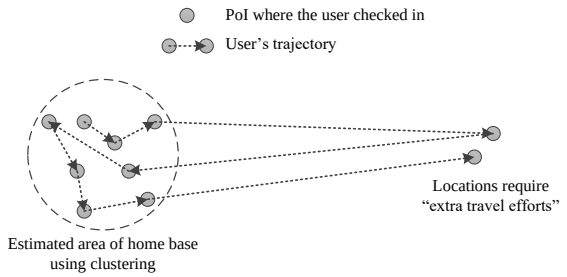

Fig. 1: Example of home base and isolated locations

For each isolated location, we estimate the travelling effort for the user to reach the location. Using the distance between the isolated location and the user's home base may be biased. A user who likes to drive can easily travel to a mall several miles away from home, but for someone who does not have a car, travelling the same distance means significantly more effort. Instead, we use the relative travelling distance. For a location l, we compute $D_l = d_l/\overline{d_{home}}$ where d_l is the minimal travel distance between l and any home base location, and $\overline{d_{home}}$ the average travel distance between locations within the user's home base.

D. Extracting the type of PoI

A user's spatial-temporal behaviour can be very different in different types of PoIs. Suppose a user gives high rate for both a coffee stand and a theatre. It is common for a user to visit the coffee stand every day or even multiple times a day, while such a visiting pattern is not likely to appear for his visiting to the theatre. Given the diversity of locations, we believe the type of different PoIs serves as an important feature in the rating prediction process.

Fortunately, in many spatial-temporal datasets (e.g., [19]), a category is given for each PoI. Nevertheless, a dataset may provide only coordinate of visited locations with no additional information. One way to infer the type of PoI in this case is to match the coordinates to PoIs using geoinfo systems such as Google Map, which provides detailed category information of PoIs.

E. Discussion

Due to limited space, we briefly discuss some other useful features that could be explored for user rating prediction.

Regional difference: User's visiting pattern to some PoIs can be region-sensitive. Consider an example of two restaurants. Restaurant A locates near a busy train station while restaurant B is in suburb region. Even if the B provides better service than A, A may attract much more visitors comparing with B due to its better location. A possible way to mitigate the problem is to add a feature that describes region of a location (e.g., "Downtown region", "Less popular region", etc.) but it requires extra geo-demographic information which cloud be hard to collect. A simpler and efficient way is to explore the relation between nearby PoIs of the same type. The reason is intuitive. For example, given several restaurants close to each other, if all of them are visited by 100 users per day, except for one restaurants which has only 10 visitors per day, it is very likely that this restaurants has a bad rating.

Scale of the PoI: It is worth mentioning that the scale of a PoI has an obvious impact on the total number of visits to it. A small convenient store, regardless of how users like it, is not likely to have even close number of visitors comparing with a large super market. However, to our knowledge, these is no convincing way that can accurately estimate scale of a PoI. We leave this for future exploration when such data is available.

Social connection between users: The prevalence of LBSNs makes it easy to learn social connects between users. If several users who are socially connected check in at a PoI at the same time, it is likely that they have a group social event at the PoI. The fact that a group of friends choose to meet at a PoI suggests most of the users in the group like the place. We may also assume a user's rating of some PoIs is affected by the rating of his friends, given that they frequently visit a similar set of PoIs. In general Group-visits may demonstrate different pattern comparing with individual visits, and therefore can be used as a distinguished feature.

VI. User-Site Rating Prediction

Given a set of user trajectories and a set of PoIs visited by these users, we aim at predicting the users' rating for each of the PoI. The rating is usually expressed as a numeric score. We assume the score is no-negative and can be continuous. A higher score corresponds to a better user rating. We propose to use out-of-the-box machine learning algorithms to training the prediction model with the aforementioned spatial-temporal features as input. In order to learn the model, we assume the accurate ratings from some users for certain PoIs are given. This information can usually be retrieved from websites that provide reviews of PoIs, such as Yelp, FourSquare, GoogleMap, and TripAdvisor.

We do not intend to explore existing machine learning techniques in an encyclopaedia way to find the most effective one for our problem. Instead, our goal is to demonstrate the spatial-temporal features of a user has the potential to predict his rating of PoIs he visited. To this end, we choose two representative techniques, **linear regression** and the standard **deep neural networks** (DNN) [16]. Linear regression is chosen due to its simplicity; Deep neural network, although is much more expensive to train, demonstrates good performance in many applications (e.g., [15], [3], [7]).

A DNN is a artificial neural network with more than one hidden layer. Each hidden layer consists of certain number of hidden units or neurons. A hidden unit i uses a output function $f(x_i) = y_i$ to map its input x_i, received from the previous layer, to an output y_i, which is then sent to the next layer. Typical output functions include linear, logistic, sigmoid, softmax, etc. In our experiments, we construct a DNN with a simple fully-connected 4-layer architecture. The DNN is discriminatingly trained by backpropagation. A cost function is used to measure the discrepancy between the target output and predicted value in backpropagating. For our problem, we use Mean Square Error (MSE) as cost function.

Since we are dealing with a relatively small number of features, here we will not concern ourselves with the problem of handling overfitting and optimizing of training time.

VII. Experimental Evaluation

A. Experiment Settings

We evaluate the proposed technique over the FourSquare check-in dataset [19]. The dataset contains 227,428 check-ins in New York city and 573,703 check-ins in Tokyo collected in 10 month. A PoI type is given for each record. We use the user ratings on FourSquare as ground truth. Specifically, FourSquare uses a 10-point rating system, but a user cannot give arbitrary points to a PoI. A user can select from three options: "Like" (count as 10 points), "Neither like nor dislike" (count as 5 points), or "Dislike" (count as 0 point). Which PoI is liked by a user is publicly available. We have implemented three schemes:

- Linear regression (LR) A baseline approach that predicts user ratings with a model generated by simple linear regression with the proposed features.
- Matrix factorization (MF) The simple non-negative matrix factorization [6] for recommender systems. We use matrix factorization to predict the rating of each users who have visited a PoI, and then use their average rating score as the "overall" rating of the PoI.
- Deep neural network (DNN) The scheme predicts user ratings with a model trained by deep neural network as proposed in Section VI, using the proposed features.

For LR and DNN, we use the four types of features discussed in Section V. More parameters of our experiment is given in Table II. Our experiment platform is a virtual machine with Intel Xeon 64-bit 8-core CPU running on 2.93GHz and 32GB RAM. The algorithms are implemented with Microsoft Azure machine learning toolkit and modules.

TABLE II: Groups based on how often user visits a location

Parameter	Default value
Number of users	5000
Number of PoIs	1500
Training set size (%)	10%
Cost function	Mean Square Error
Neural network depth (for DNN)	4
Neurons per layer (for DNN)	12
Output function (for DNN)	sigmoid
Number of latent features (for MF)	8

B. Data Cleaning

A few data cleaning steps are needed to prepare the dataset. First, we remove the *inactive* users from the dataset. We define inactive users as those who have less than 1 check-ins per week in the 10 month period, and have less than 10 liked locations. The lack of data makes it hard to extra features for these users. Second, we remove the PoIs which were not rated by any of these users, since the ground truth is not available for such PoI. After the cleaning process, we extract the check-in information of approximately 5000 most active users and 1500 PoIs they have visited from the dataset.

In order for MF to work, we also generate a user-PoI rating matrix based on the users and PoIs appeared in the FourSquare dataset. The user ratings are directly crawled from FourSquare.

C. Prediction Results

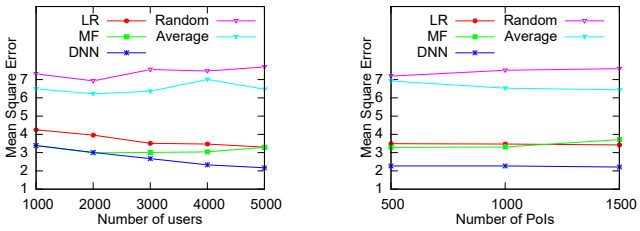

Fig. 2: MSE of different prediction schemes

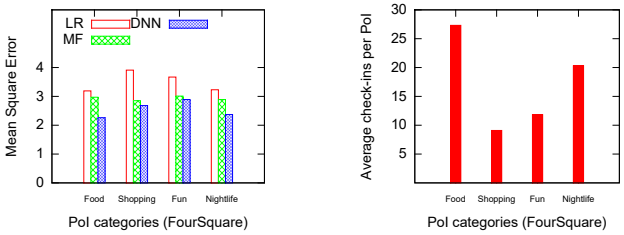

Fig. 3: MSE of different PoI categories

We compare the Mean Square Error (MSE) of rating prediction results of the three schemes. We adjust the number of users and PoIs, respectively, and exam the performance trend of the schemes. For comparison, we also calculate the MSE of two baseline approaches: **Random** and **Average**. Random randomly generate a score in $[0, 10]$ as user rating for each PoI, while Average uses the average rating of all the PoIs in the dataset as its prediction. The result is plotted in Figure 2. The

proposed DNN prediction method demonstrate clear advantage over the other schemes for a moderate number of users. The performance of LR is also close to that of pure matrix factorization for larger number of users. The performance of DNN increases as the number of users increases. This implies a larger group of users overall have more reliable spatial-temporal features. In contrast, the performance of MF drops as the number of users/PoI increases. As the size of user-PoI rating matrix increases, it becomes more spares, making it harder for MF to generate accurate predicts.

We also shows the MSE of the prediction result for PoIs in four different categories, namely "Food", "Shopping", "Fun", and "Nightlife". These PoI categories are provided by FourSquare. There appears to be a difference of prediction accuracy between PoI categories (Figure 3-a). For the proposed methods, the prediction result for "Food" and "Nightlife" shows a lower MSE comparing with that of "Shopping" and "Fun". A closer look at check-in data in different PoI category reveals that users reported less check-ins for "Shopping" and "Fun" PoIs comparing with that of 'Food" and "Nightlife" (Figure 3-b). Thus the relatively worse performance can be explained by the lack of data, which can affect the quality of extracted features.

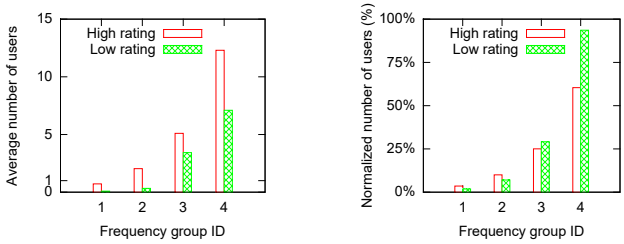

Fig. 4: User frequency group distribution

Figure 4 compares PoIs with different ratings in terms of the (normalized) average number of user in each frequency group, as showed in Table I. We define high rating PoI as a PoI with a overall score of 7.5 or higher out of 10. While a low rating PoI is one with an overall score of 2.5 or less. In general, highly rated PoIs have more visitors. Furthermore, we observe that a highly rated PoI attracts more frequent-visitors among all the visitors, comparing with PoIs with low ratings. These results support the assumptions we made in Section V.

VIII. Conclusions

Location-based social networks such as FourSquare, Yelp, TripAdvisor and OpenRice allow users to "check-in" and rate sites such as restaurants and hotels, to quantify their experience with that site. However, the vast majority of users does not use such online platforms to publish their opinion. To improve such user-site recommendation systems, we propose to exploit spatio-temporal data to estimate user-site ratings of any user. Therefore, we analysed trajectory data to find discriminative features indicating user preferences, such as their frequency of visit of a site, duration of visit, and distance from their home base. Using these features, we use existing (explicit) user-site recommendations to learn the relationship between our proposed features and the this ground-truth. Our experiments show, that our solution allows to drastically reduce the user-site-rating prediction error by exploiting spatio-temporal data. To leverage our solution to large spatio-temporal datasets, our next step is to automatically detect stay-points, rather than using pre-labelled check-in data.