# Personalized Visited-POI Assignment to Individual Raw GPS Trajectories

Jun Suzuki<sup>\*</sup> NTT Communication Science Laboratories, NTT Corporation Yoshihiko Suhara<sup>†</sup> Hiroyuki Toda NTT Service Evolution Laboratories, NTT Corporation Kyosuke Nishida NTT Media Intelligence Laboratories, NTT Corporation

## Abstract

Knowledge discovery from GPS trajectory data is an important topic in several scientific areas, including data mining, human behavior analysis, and user modeling. This paper proposes a task that assigns personalized visited-POIs. Its goal is to estimate fine-grained and pre-defined locations (i.e., points of interest (POI)) that are actually visited by users and assign visited-location information to the corresponding span of their (personal) GPS trajectories. We also introduce a novel algorithm to solve this assignment task. First, we exhaustively extract stay-points as candidates for significant locations using a variant of a conventional stay-point extraction method. Then we select significant locations and simultaneously assign visited-POIs to them by considering various aspects, which we formulate in integer linear programming. Experimental results conducted on an actual user dataset show that our method achieves higher accuracy in the visited-POI assignment task than the various cascaded procedures of conventional methods.

## 1. Introduction

The availability of personal spatial-temporal data continues to rapidly increase. This is because personally equipped mobile devices such as smartphones have become ubiquitous and are generally equipped with GPS devices that can constantly record the trajectories of latitude-longitude positions. This situation provides opportunities to discover valuable knowledge from such personal spatial-temporal data. In fact, knowledge discovery tasks from GPS trajectory data have already been proposed, such as predicting movement destinations [3, 38], recommending points of interest (POIs) around locations [20, 21, 32, 37, 40], and recognizing individual mobility [42, 44]. Unfortunately, mining knowledge from raw GPS trajectory data is not a straightforward task because such data are merely a series of real-valued positions and timestamps. We need to link the GPS trajectory data to the actual world's semantic information, including that for locations, events, transportation modes, user activities, and personal preferences. Hence, GPS trajectory mining is attracting a great deal of attention in the geospatial data mining community.

<sup>\*.</sup> Current affiliation: Tohoku University

<sup>†.</sup> Current affiliation: Megagon Labs

Following this line of study, we discuss a similar but new GPS trajectory data mining challenge called a personalized visited-POI assignment task. Its goal is to estimate the fine-grained locations that a user actually visits and assign the user's (personal) GPS trajectories to them. A visited-POI indicates not only location history but also user habits and preferences. At least two crucial applications can be inferred from personalized visited-POI assignment tasks. One is the automatic construction of a user's personal lifelogs from his/her personal spatial-temporal data based on GPS trajectories. Since annotating all GPS trajectory data by hand is unmanageable because the amount of GPS trajectory data is excessive, a personalized visited-POI assignment system might significantly reduce the annotation cost of constructing personal spatial-temporal lifelogs. Another application is POI recommendation systems [32, 37, 40], which basically use personal POI histories and general POI patterns that can be essentially estimated by our visited-POI assignment method.

The automatic assignment of personalized visited-POIs to individual raw GPS trajectories is challenging. For example, several researchers have tackled the task of understanding user-specific activity from raw GPS trajectory data [16, 19, 44]. In their work, they assumed that a *stay-point*, which is a spot where the user remains over a constant time-span within a certain area, is a unit that pertains to a meaningful location of a user. Since they assume that stay-points are accurately extracted, detecting significant locations from them remains an open question. This is because the stay times of significant locations vary, and a stay-point extraction algorithm with just one fixed parameter cannot extract all of the significant locations without errors.

We propose a novel framework to solve this task. We first enumerate all of the possible candidates of the time-spans for a visited-POI using an approach that resembles the conventional stay-point extraction method with a very conservative parameter setting. Our approach extracts the true time-spans at which the user actually stayed in the visited-POIs from a GPS trajectory with very high recall. Thus, we can select the significant time-spans from the extracted stay-points and simultaneously assign their visited-POIs. At that time, we need to capture several different aspects of the personal and general human behavior for accurate estimation. For example, we must consider the validity of the extracted stay-points, the likelihood of the visited-POI assignments, the validity of the visited-POI relations, and the validity of the visited-POI sequence length. To simultaneously take these heterogeneous variables into account, we formulate the challenge as an instance of a combinatorial optimization problem and solve it as integer linear programming (ILP).

Note that one aspect about which we must be cautious is the task settings that are related to how we obtain and incorporate personal preferences and information. People are often unwilling to upload their GPS trajectories (justifiably so) and their location (visited-POIs in our case) histories to servers because of privacy concerns [36]. Thus, we assume that users' personal information, including their visited-POI histories, cannot be aggregated in one system of a service. Instead, we assume that individual GPS histories are collected in a user's own storage. This means that the personal information of other users cannot be incorporated for building a visited-POI assignment system. We restrict our reach to a user's personal information for estimating his/her visited-POI assignments. We believe this is a very realistic task setting.

The following are the contributions of this paper:



Figure 1: Problem setting in this paper

- 1. We revisit and evaluate the conventional stay-point extraction algorithm and provide an appropriate strategy adjusted to the personalized visited-POI assignment task.
- 2. We propose a general framework that efficiently and accurately detects significant time-spans that can be assigned to visited-POIs.

The rest of the paper is organized as follows: We first describe the definitions and the problem setting of our personalized visited-POI assignment task in Section 2 and discuss related work in Section 3. We present the proposed method to tackle the personalized visited-POI assignment task in Section 4 and describe our experiments and results in Section 5.

## 2. Definition of Personalized Visited-POI Assignment

This section describes an overview and the definition of our proposed personalized visited-POI assignment task.

#### 2.1 Preliminary: Terms and Notations

For clarity, we first introduce some technical terms and provide their definitions used in this paper.

**Track-point**: We refer to a single data point obtained from a positioning system such as GPS as a track-point and assume that a track-point consists of a triplet: the longitude, the latitude, and the actual timestamp to be recorded. Formally, let lng, lat and ts respectively be the longitude, the latitude, and the actual timestamp. Then, a track-point G is written as G = (lng, lat, ts).

**Session**: A session is a sequence of continuous track-points. Let  $G_t$  and  $G_{t+1}$  represent successive track-points at times t and t + 1, respectively. Then session S is defined as  $S = (G_1, G_2, \ldots, G_T)$ , where T represents the number of track-points in the session. Our study obtains one session from a sequence of GPS trajectory for a day, where a session always starts at 0:00 midnight and ends at 23:59:59 to simplify the task definition. **Stay-point**: A stay-point [12, 45] is a sub-sequence of the session S at which the user remains over a certain pre-defined period in the same area. Formally, a stay-point sp consists of successive track-points in a session S, namely,  $sp = (G_i, G_{i+1}, \ldots, G_j)$ , where i and jare the indices of the starting and ending points of the stay-point, that is,  $1 \le i \le j \le T$ . Hereafter, the relation between a sub-sequence A in a sequence B is denoted as  $A \subseteq B$ . Thus, the relation between a stay-point sp in the session S can be written as  $sp \subseteq S$ .

In this paper, each stay-point has five additional attributes, namely, the center location (lng, lat) of the sequence of the track-points in the stay-point, start timestamp bt, end timestamp et, and stay-time st, all of which are easily calculated from the sequence of track-points in the stay-point. Suppose  $sp_k = (G_i, G_{i+1}, \ldots, G_j)$  represents the k-th stay-point in a session S. We refer to the center locations of the longitude, the center location of the latitude, the start timestamp, the timestamp, and the stay-time of  $sp_k$  as  $sp_k.lng$ ,  $sp_k.lat$ ,  $sp_k.bt$ ,  $sp_k.et$ , and  $sp_k.st$ , respectively;

$$sp_k.lng = \frac{1}{j-i+1} \sum_{t=i}^{j} G_t.lng,$$
(1)

$$sp_k.lat = \frac{1}{j-i+1} \sum_{t=i}^{j} G_t.lat,$$
 (2)

$$sp_k.bt = G_i.ts,$$
(3)

$$sp_k.et = G_j.ts,$$
(4)

$$sp_k.st = G_j.ts - G_i.ts.$$
(5)

**Point of Interest (POI)**<sup>1</sup>: A POI [20, 32, 37, 39, 40] is a place in which people may be interested. Suppose we have a pre-defined set of POIs (POI database) denoted as  $\mathcal{P}$ . This paper defines that each POI consists of the following four attributes; its own name, its category, and its location (longitude, latitude) information. Let  $poi_k$  represent the k-th POI in  $\mathcal{P}$ , that is,  $poi_k \in \mathcal{P}$ . We refer to the name, category, longitude and latitude of  $poi_k$ as  $poi_k.name$ ,  $poi_k.cat$ ,  $poi_k.lng$  and  $poi_k.lat$ , respectively.

Then we define two technical terms, *visited-POI* and *significant location*, that are the keys for describing our proposed task.

**Visited-POI**: A visited-POI is a POI that the user has actually visited in the real world. Note that a visited-POI also consists of identical attributes as POI, namely, the name, category, longitude and latitude.

Significant location: Similar to the stay-point, a significant location is a sub-sequence of session S while the user actually visited to a POI. In other words, this is a sequence of track-points in S that can be assigned to a corresponding visited-POI. We define that significant location also consists of identical attributes as a stay-point. This means that each significant location also has five additional attributes; the center location of the track-points, the start timestamp, the end timestamp, and the stay-time of the significant location as well as stay-point.

<sup>1.</sup> See https://en.wikipedia.org/wiki/Point\_of\_interest for a quick reference.

## 2.2 Task Definition and Assumptions

In this section, we propose a task called *personalized visited-POI assignment task*. With the terms and definitions explained in the previous section, this section describes our task definition. Fig. 1 shows an overview of our task.

We first explain a few required assumptions for setting our task a meaningful study. The first assumption addresses about the relation between stay-point and significant location.

**Assumption 1.** We assume that each significant location is always a stay-point.

This assumption means that we infer that users always stay for a certain duration of time if they visit a POI. In other words, the task proposed in this paper only considers the POIs at which the users stay for a particular period of time, i.e., for one minute. This assumption is not unrealistic since users generally have their own purposes to do for *visiting* a POI. Note that the above assumption does not mention that a stay-point is always a significant location. For example, the situations of being stuck in traffic, waiting at railroad crossings, and talking on cell phones in open spaces without changing locations are typical and intuitive examples of stay-points that are not significant locations. In addition, Fig. 1 also illustrates the relation among stay-points, significant locations, and visited-POIs, as defined in the previous section.

The second assumption is about the pre-defined POI database used in our paper.

Assumption 2. We assume that a (common) POI database  $\mathcal{P}_c$  is obtained from a conventional location-based service (e.g., Foursquare<sup>2</sup>). We also assume that each user may have his/her own personal POI database  $\mathcal{P}_p$ , which includes his/her home, office, and places in which he/she is specifically interested.

This assumption indicates that all users might have their own POI databases  $\mathcal{P}_{p+c} = \mathcal{P}_p \cup \mathcal{P}_c$ . Note that if user does not have his/her own POI database  $\mathcal{P}_p$ , then  $\mathcal{P}_{p+c} = \mathcal{P}_c$ .

The following is an assumption about how we determine the significant locations and visited-POIs.

**Assumption 3.** In this paper, we assume that the manually annotated visited-POIs and the significant locations by users themselves are true visited-POIs and significant locations.

This assumption is derived because precisely identifying the places actually visited by users is difficult. Therefore, we trust the annotations of users. Moreover, since the granularity of POI annotation deeply depends on users, significant locations also deeply depend on users intentions about where they believe they visited.

The following is the assumption about the relation among significant locations.

**Assumption 4.** The duration of visits, in other words, the duration of significant locations, never overlaps.

This assumption is reasonable and realistic since a user cannot physically visit more than two places at one time. This assumption reflects our annotation scheme; users cannot annotate overlapping visited-POIs and significant locations since they are not expected to

<sup>2.</sup> https://foursquare.com/

simultaneously visit more than two POIs. Given two stay-points  $sp_m$  and  $sp_n$  in a session S, namely,  $sp_m \subseteq S$  and  $sp_n \subseteq S$ ,  $sp_m$  and  $sp_n$  are disjoint (non-overlap) stay-points in terms of the duration of the stays if the relation  $G_a \neq G_b$  holds for all  $G_a \in sp_m$  and  $G_b \in sp_n$ . We represent the relation of such non-overlapping two sub-sequences as  $sp_n \cap sp_m = \emptyset$ .

**Personalized visited-POI assignment task**: Let  $(sp_n, poi_n)$  represent the *n*-th significant location with corresponding visited-POI, where  $sp_n$  and  $poi_n$  represent the *n*-th significant location and its corresponding visited-POI, respectively. Then given a (personal) POI database  $\mathcal{P}_{p+c}$  and a session S, the personalized visited-POI assignment task is to find  $\{(sp_n, poi_n)\}_{n=1}^M$  under a condition of  $sp_{n_1} \cap sp_{n_2} = \emptyset$  for all  $n_1, n_2 \in \{1, \ldots, M\}$  and  $n_1 \neq n_2$ , where  $poi_n \in \mathcal{P}_{p+c}$ , and  $sp_n \subseteq S$ . Moreover, M represents the number of true visited-POIs obtained from given  $(\mathcal{P}_{p+c}, S)$ . Therefore, the input and output of our task are  $\mathcal{I} = (\mathcal{P}_{p+c}, S)$  and  $\mathcal{O} = \{(sp_n, poi_n)\}_{n=1}^M$ , respectively.

## 3. Related Work

This section explains related work in terms of several perspectives of our proposal.

## 3.1 GPS trajectory mining and stay-point extraction

Many studies on GPS trajectory mining exist, such as user activity estimation [8, 11, 22, 23], transportation mode detection [43, 46], and region analysis [35, 39]. A typical approach to tackle these tasks first extracts stay-points as a clue for solving them. Therefore, we believe that stay-point extraction is a key technology of many GPS trajectory mining tasks.

Various stay-point extraction methods have already been proposed. For example, Ashbrook and Starner [2, 3] use a modified k-means method, Adams et al. [1] use DBSCAN [10], and Kurashima et al. [17] employ Mean-Shift [7], all of which are based on clustering. Kang et al. [12] and Zheng et al. [45] assume that stay-points are positions within a constant radius from a center where the stay time exceeds a constant time. More recently, Nishida et al. [30] developed a more robust stay-point extraction algorithm that considers outliers and missing points in GPS trajectories.

As in other GPS trajectory mining tasks, we can leverage a stay-point extraction method to obtain the clues of visited-POIs in our visited-POI assignment task. However, it is nearly impossible to extract only meaningful stay-points for the downstream task that we are actually aiming to solve (visited-POI assignments) by simply applying a conventional staypoint method. This is because the above stay-point extraction methods were developed independently of downstream tasks, and thus they extract stay-points regardless of the objectives of the downstream tasks. Thus, treating all the identified stay-points as if they were all visited-POIs does not serve the purpose of our task.

#### 3.2 Detecting semantic locations

The challenge that most resembles our personalized visited-POI assignment task is detecting semantic locations from GPS trajectory data [5, 24, 45]. Cao et al. [5] extracted stay-points from trajectories and combined them with street addresses obtained by a reverse geocoder. Their method assigns a semantic label to stay-points by yellow-pages. This strategy resembles a nearest neighbor assignment to stay-points by a POI database. Liu

et al. [24] also extracted semantic locations from GPS trajectories in the same manner. Zheng et al. [45] extracted stay-points from user trajectories and applied a hierarchical clustering algorithm to combine stay-points to create hierarchical stay areas on a diagram called a tree-based hierarchical graph. The key difference between our personalized visited-POI assignment task and a semantic location detection task is that the semantic location is essentially determined on the basis of stay-points, while in this paper we determine a visited-POI on the basis of whether the user actually visits it.

#### 3.3 POI recommendations

POI recommendation tasks are closely related to our target task. Many previous studies have addressed POI recommendations [4, 18, 33, 37, 40]. Most used the collaborative filtering (CF) approach, which requires inter-user information, to achieve recommendations. Leung et al. [18] performed co-clustering on users and stay-points to improve CF recommendations. Ye et al. [37] proposed a framework that fuses user preferences to a POI with both social and geographical influences. Wang et al. [33] showed that the most frequently used check-in history in location-based social networks is first-visit POIs and proposed a personalized PageRank-based method to improve the accuracy of estimating first-visit POI recommendations. Yuan et al. [40] introduced a time-aware feature into a CF-based approach and showed that incorporating temporal and spatial influences improves the accuracy of POI recommendations. These studies exploit other user check-in histories to recommend POIs to users. Bao et al. [4] studied location recommendation with a location category hierarchy and concluded that since different users have varying levels of expertise and preferences, they should be treated differently in the recommendation process.

These CF-based methods used in the POI recommendation task resemble our method. However, they require the histories of many users; our target task, which is a personalized visited-POI assignment, only assumes using individual GPS trajectories and the annotation data of users. Therefore, since these methods are related without being direct competitors of our proposed method, we ignore their evaluations in our experiments.

Other studies on a location naming task [20, 21] and a POI recommendation task [32] use a supervised learning algorithm [6, 25] to build POI ranking models. To formalize their problems as a ranking challenge, they look at a location (i.e., longitude and latitude) as a query and a user's check-in data to it as relevant labels. Their methods utilize user history, the statistics of POIs in a check-in service, and other information to generate features. Then the ranking model uses the features to rank POI candidates. The key difference between a visited-POI assignment task and a POI recommendation task is the latter's requirement for significant location extraction. Previous POI recommendation studies assume that significant locations are given, but our visited-POI assignment task does not. One straightforward approach is cascading a stay-point extraction algorithm and a POI recommendation method. We regard the nearest neighbor method [5] and learning-to-rank methods [20, 32] as similar approaches to our proposed method<sup>3</sup>.

<sup>3.</sup> We do not consider the method by Lian et al. [21] to be a similar method because it requires check-in histories of many users to calculate latent topic features. Its other part is equivalent to another previous work [20].

## 3.4 Other related tasks

More recently, several novel tasks have been proposed that are related to visited-POI assignments. For example, Lu et al. [26] characterized the life cycle of POIs and investigated the POI evolution process over time. Espin-Noboa et al. [9] tackled a task that clusters users based on spatio-temporal dimensions with a non-negative tensor factorization method. They clearly focus on different targets.

Zarezade et al. [41] focused on the periodic behaviors of users and formalized POI checkin patterns as a stochastic point process. An interesting aspect of their method is that they take into account a factor of the influence of the close friends of users. In contrast, our task detects actual visited-POIs from obtained raw GPS trajectories and POI information, which includes the user's periodic behaviors without being limited to them. Therefore, our task indirectly includes their task, even though it does not specifically focus on periodic behaviors.

Wang et al. [34] proposed a sequential personalized spatial item recommendation framework (SPORE), which recommends a sequence of POIs based on individual POI-visit histories. Their target closely resembles ours. However, the essential difference is that their task assumes a sequence of check-in records as input, unlike raw GPS trajectories for our case. This means that their method does not assume that an input sequence (check-in records) contains any false positive information, which is one of the main challenges of our task. In addition, SPORE, their proposed algorithm, cannot be directly applied to GPS trajectories since it does not have a mechanism that removes false positive stay-points, while our method can remove such meaningless stay-points.

Lv et al. [27] proposed a task that detects personally semantic places from GPS trajectories. Their proposed task also appears to closely resemble ours. However, their target is to detect places (*frequently* visited by an individual user) that might have such important semantic meanings as home or office. In this perspective, their target is closely related to Zarezade et al. [41], as explained above. In contrast, our proposed task detects not only frequently visited places like homes and offices but also every POI that the user actually visits regardless of the frequency.

Keles et al. [13] employed a Bayesian network to detect the categories of visited-POIs, such as hospitals and universities, from the GPS trajectories of vehicles. Their motivation is closely related to ours. The essential difference is that they only detect the categories of visited-POIs; we detect the visited-POIs themselves. Additionally, they used vehicles' GPS trajectories, whereas we target the trajectories obtained from the mobile devices of users. Thus, our challenge is much more complicated.

As discussed above, motivation, goals, and task settings of all the studies differ from our task even though all of these proposed tasks are related.

#### 4. Joint Estimation of Significant Locations and Their Visited-POIs

In this section, we describe how we model and solve the proposed task, personalized visited-POI assignment task. We first discuss the properties that our model must provide and suggest a mathematical formulation that implements our proposal. We also elaborate on the features needed to capture user behaviors. Finally, we explain the procedure through



Figure 2: Example of long-distance and higher-order dependency

which our model simultaneously extracts a significant location and its visited-POI from raw GPS trajectories.

## 4.1 Two-step approach: Simplification for practical modeling

It is computationally infeasible to consider all the durations of the track-points as candidates of significant location for assigning visited-POIs. To avoid such computational difficulty, we utilize a two-step approach to derive an effective algorithm for personalized visited-POI assignments. The following is a brief explanation of our two-step procedure:

- 1. Extract stay-points using stay-point extraction with a high-recall setting, and collect the visited-POI candidates for each extracted stay-point.
- 2. Evaluate whether each extracted stay-point is a visited-POI.

As we described in Section 3.1, many previous methods applied to GPS trajectory mining tasks leverage a stay-point extraction method as a first-step process to obtain clues about the target mining tasks. We also follow this approach for our personalized visited-POI assignment task. Note that the stay-point extraction method is not limited to any specific scheme.

# 4.2 Required properties for modeling

The remaining part of our task is selecting significant locations from stay-points and assigning a visited-POI to each one. We discuss the requirements for an ideal method of selecting significant locations and assigning a visited-POI to them. In our approach, the method must be able to detect a significant location from an exhaustive amount of stay-points. For example, as explained in Section 2, the method must determine whether each stay-point is meaningless (such as traffic jams) or a significant location. To successfully do so, we need to simultaneously consider different aspects to accurately solve our task, for example, estimating the relation between a significant location and its visited-POI. We also need to satisfy the consistency of the significant location of each estimated visited-POI. This is because the significant location of any two visited-POIs never overlaps, and such higher level knowledge as POI-POI interactions might provide very useful information for solving our task. According to the above analyses, a Hidden Markov Model (HMM) [29] and a Conditional Random Field (CRF) [29] are possible candidates for modeling the personalized visited-POI assignment task. This is because a k-th order Markov model (HMM/CRF) can capture k-consecutive patterns of visits and is efficiently solvable through dynamic programming (DP).

However, the task's essential information is often retained in the long-distance (nonconsecutive) and higher-order dependency information, which cannot be represented by k-consecutive patterns of visits. For example, consider a situation where a father drops off his daughter at day-care on his way to work in the morning, moves around various offices during the day, and finally picks her up after work in the evening. Fig. 2 illustrates this example. In this example, if the father drops his daughter off at day-care on his way to work in the morning, then he must pick her up after work on his way back home with very high probability. In contrast, he does not have to visit the day-care before returning home if he did not take his daughter to day-care in the morning. Therefore, the following order of four POIs can be a visiting-pattern block for him: (home)-(day-care)-(day-care)-(home). The important point here is that this pattern doesn't have to be consecutive; he might (randomly) visit other places, such as a restaurant, a drug store or a museum, between the day-care center and his home, regardless of the pattern of visits. This implies that a user's subsequent visits do not always depend only on the last (consecutive) visited-POI; they often depend on the visited-POIs long before the user visited them. Thus, we aim to include all the combinatorial dependencies among all the stay-points and the POIs. However, there is dilemma that such long-distance and higher-order relations cannot be efficiently calculated by the DP in the k-th order Markov model (HMM/CRF) in practical k. In fact, DP's calculation cost may become identical as that of a naive exhaustive search if we consider all of the long-distance and higher-order dependency information. Such a naive exhaustive search is infeasible if the number of candidates (number of stay-points and POIs) is too large.

As an alternative for much better modeling than HMM/CRF, we formulate our task as a 0-1 integer linear programming (0-1 ILP) problem. An ILP is a good choice to efficiently solve exhaustive searches or even NP-hard problems. A general ILP solver, which is a package that aggregates many mathematical optimization techniques, discards non-optimal solutions to reduce the search cost based on the constraints of an ILP problem. The search cost depends on a problem's difficulty. In this paper, we verify that our problem can be solved by a general ILP solver in a reasonable time. ILP formalization is a general form that embraces both HMM and CRF [31]. Since a special case of ILP formalization perfectly conforms to HMM/CRF, we emphasize that our framework includes HMM/CRF.

## 4.3 Modeling by 0-1 ILP formulation

Table 1 summarizes all the notation symbols and their descriptions for our formulation. All are binary (0-1) variables.  $s_i$  and  $\bar{s}_i$ , which are assigned to each stay-point, denote variables that indicate whether a location is significant.  $(s_i = 1, \bar{s}_i = 0)$  if the *i*-th stay-point is a significant location, and  $(s_i = 0, \bar{s}_i = 1)$  otherwise.  $v_{ik}$  represents a variable for visited-POIs assigned to each visited-POI candidate.  $v_{ik} = 1$ , if the *k*-th POI is the visited-POI of the *i*-th stay-point, and  $v_{ik} = 0$  otherwise.  $t_{ijkl}$  denotes the interaction of two visited-POIs

Symbol	Description					
$s_i, ar{s}_i$	Pair of variables that represent whether the <i>i</i> -th stay-point is a signifi-					
	cant location $(s_i = 1, \bar{s}_i = 0)$ or not $(s_i = 0, \bar{s}_i = 1)$ .					
$v_{ik}$	Variable that represents whether the $k$ -th visited-POI candidate at the					
	<i>i</i> -th stay-point is a visited-POI $(v_{ik} = 1)$ or not $(v_{ik} = 0)$ .					
$t_{ijkl}$	Variable that represents whether the k-th visited-POI candidate in the					
	i-th stay-point and the $l$ -th visited-POI candidate in the $j$ -th stay-point					
	are both visited-POIs $(t_{ijkl} = 1)$ or not $(t_{ijkl} = 0)$ .					
$y_m$	Variable that represents whether the number of visited-POIs is $m (y_m =$					
	(1) or not $(u_m = 0)$ .					

Table 1: Notation for our formulation



Figure 3: Diagrammatic illustration of variables in our method

with their stay-points.  $t_{ijkl} = 1$ , if the k-th POI is the visited-POI of the *i*-th stay-point, and the *l*-th POI is the visited-POI of the *j*-th stay-point, and  $t_{ijkl} = 0$  otherwise.  $y_m$  also represents the visited-POI sequence length.  $y_m = 1$ , if the sequence length is *m*, and  $y_m = 0$  otherwise.

Fig. 3 illustrates the formalization as an 0-1 ILP problem. A lower circle node denotes an extracted stay-point, and an upper circle node surrounded by a rectangle (plate) denotes a candidate of the corresponding visited-POI. A plate represents node replicates, and  $K_i$ denotes the number of visited-POI candidates at the *i*-th stay-point. We also used visited-POI interaction variable  $t_{ijkl}$  that corresponds to the combination of the *k*-th visited-POI candidate at the *i*-th stay-point and the *l*-th visited-POI candidate at the *j*-th stay-point. Let S,  $\overline{S}$ ,  $\mathcal{V}$ ,  $\mathcal{T}$ , and  $\mathcal{Y}$  respectively represent a set of all the variables of  $s_i$ ,  $\overline{s}_i$ ,  $v_{ik}$ ,  $t_{ijkl}$ , and  $y_m$ . We formalize the joint estimation task as 0-1 ILP problem P as follows:

$$P: \max_{\mathcal{S}, \bar{\mathcal{S}}, \mathcal{V}, \mathcal{T}, \mathcal{Y}} \qquad \sum_{i} \left\{ \langle \mathbf{w}^{(s)}, \mathbf{x}^{(s_i)} \rangle s_i + \langle \mathbf{w}^{(\bar{s})}, \mathbf{x}^{(\bar{s}_i)} \rangle \bar{s}_i \right\}$$
(6)

$$+\sum_{i}\sum_{k} \langle \mathbf{w}^{(v)}, \mathbf{x}^{(v_{ik})} \rangle v_{ik}$$
(7)

$$+\sum_{i}\sum_{j}\sum_{k}\sum_{l}\langle \mathbf{w}^{(t)}, \mathbf{x}^{(t_{ijkl})}\rangle t_{ijkl}$$
(8)

$$+\sum_{m} \langle \mathbf{w}^{(y)}, \mathbf{x}^{(y_m)} \rangle y_m \tag{9}$$

subject to:  $s_i + \bar{s}_i = 1, \sum_{(i,j) \in L} s_i + s_j \le 1 \quad \forall i$  (10)

$$\sum_{k} v_{ik} = s_i \ \forall i \tag{11}$$

$$\sum_{k} \sum_{l} t_{ijkl} \le 1 \ \forall i, j \tag{12}$$

$$t_{ijkl} \le v_{ik}, \ t_{ijkl} \le v_{jl} \ \forall i, j, k, l$$
(13)

$$\sum_{m} y_m = 1, \ \sum_{m} m y_m = \sum_{i} s_i.$$

$$(14)$$

Here,  $\mathbf{x}^{(\cdot)}$  denotes a feature vector that corresponds to variable (·), and weight vector  $\mathbf{w}^{(\cdot)}$  denotes a weight vector for feature  $\mathbf{x}^{(\cdot)}$ .  $\langle \cdot, \cdot \rangle$  denotes the inner product of the given vectors. In Eq. (10),  $L = \{(i, j) | i, j \in I, i \neq j, \text{ol}(sp_i, sp_j)\}$ , where I denotes the set of stay-point indexes, and  $\text{ol}(sp_i, sp_j)$  returns true if stay-points i and j overlap.

The objective function consists of the following four parts: (1) stay-point features, (2) stay-point and POI features, (3) POI-POI features, and (4) sequence length features. Each feature value is calculated using a user's annotated data. Weight vectors  $\mathbf{w}^{(\cdot)}$  are also estimated from the user's annotated data. Note here that training the features and the weights are the personalized parts in our approach since we basically train them just using a single user's personal data. We expect that such training can capture some personalized behaviors in the features and the weights.

After we describe the details and the calculation methods of these features, we explain the details of the constraints of problem P.

#### 4.4 Features and constraints in ILP formulation

This section describes the features and constraints in Eqs. (6) to (14).

#### 4.4.1 Stay-point features

Stay-point features  $\mathbf{x}^{(s_i)}$  and  $\mathbf{x}^{(\bar{s}_i)}$  in Eq. (6) show the validity and the invalidity of staypoint *i* as a significant location. Higher values in  $\mathbf{x}^{(s_i)}$  indicate that it is more reasonable to choose stay-point *i* as a significant location ( $s_i = 1$ ). In this paper, we prepare two features,  $\mathbf{x}^{(s_i)} = (x_1^{(s_i)}, x_2^{(s_i)})$ , to capture the validity of a stay-point. As for  $\mathbf{x}^{(\bar{s}_i)} = (x_1^{(\bar{s}_i)}, x_2^{(\bar{s}_i)})$ , we use  $x_1^{(\bar{s}_i)} = 1 - x_1^{(s_i)}$  and  $x_2^{(\bar{s}_i)} = 1 - x_2^{(s_i)}$  to create features.

For  $x_1^{(s_i)}$ , we use a Gaussian basis function to utilize a user's significant location history information. This function calculates the value based on the center of stay-point *i* and the nearest stay-point in the annotated data in the following equation:

$$x_1^{(s_i)} = \exp\left\{-\gamma \parallel sp_i - \operatorname{NN}_{\operatorname{train}}(sp_i) \parallel_2^2\right\}.$$

Here,  $\gamma$  is an accuracy parameter of the Gaussian basis function, which controls the decreasing degree of the function value. If  $\gamma$  has a large value, it decreases more rapidly with distance, and if it has a small value, it decreases less rapidly.  $NN_{train}(\cdot)$  returns a stored significant location whose center is the nearest to the input stay-point.  $|| sp_i - sp_j ||_2$  calculates the geographical distance between  $sp_i$  and  $sp_j$ .

As a second feature,  $x_2^{(s_i)}$ , we model the validity of stay-point *i* based on its stay time to capture the natural characteristic that a longer time stay implies a higher probability as a significant location. We use the cumulative probability function of the exponential distribution to calculate the feature values. The distribution describes the time between events in a process in which events occur continuously and independently at a constant average rate. The cumulative probability function is calculated by

$$x_2^{(s_i)} = 1 - \exp(-\lambda s p_i \cdot s t),$$

where  $\lambda$  denotes a parameter of the average event number per unit time. The parameter can be manually fixed (e.g.,  $\lambda = 1/30$  if the average interval time is 30 minutes) or estimated using individually annotated data.

#### 4.4.2 Stay-point and POI-interaction features

The second part of the objective function represents the validity of visited-POI k at staypoint *i*. Suppose each POI generally has a *POI category* that is preliminarily defined by a thesaurus, such as home, train station, or office. The POI category plays a critical role when we estimate the visited-POIs for roughly capturing personal behaviors in the model. We used three features for  $\mathbf{x}^{(v_{ik})}$ : (1) a feature based on a user's visited-POI histories, (2) a user's visited-POI category histories, and (3) POI-category stay-time information.

For a user's visited-POI history, we use multinomial distribution to model the historical information:

$$x_1^{(v_{ik})} = \alpha_1 f_{\text{poi}}^{\text{time}}(k, sp_i.bt) + (1 - \alpha_1) f_{\text{poi}}(k),$$
(15)

where  $f_{\text{poi}}(\cdot)$  denotes the probability of visited-POI k, which is estimated based on the ratio of POI k in the user's annotated data.  $f_{\text{poi}}^{\text{time}}(\cdot, \cdot)$  denotes the time-dependent version of  $f_{\text{poi}}(\cdot)$ : probability of visited-POI k in  $sp_i.bt$ . We split a day's 24 hours into four sixhour windows to calculate  $f_{\text{poi}}^{\text{time}}(\cdot, \cdot)$ . We also prepared a feature based on POI-category information in a similar way:

$$x_2^{(v_{ik})} = \alpha_2 f_{\text{cat}}^{\text{time}}(\text{cat}(k), sp_i.bt) + (1 - \alpha_2) f_{\text{cat}}(\text{cat}(k)),$$

where  $\operatorname{cat}(k)$  denotes the category of POI k and  $f_{\operatorname{cat}}^{\operatorname{time}}(\cdot, \cdot)$  and  $f_{\operatorname{cat}}(\cdot)$  are the category versions of the functions in Eq. (15). Here  $\alpha_1, \alpha_2 \in [0, 1]$  are the smoothing parameters for the linear interpolation of the time independent probability functions.

The third feature,  $x_3^{(v_{ik})}$ , is calculated using the stay-time likelihood of the POI category. We assume that each POI category has its own particular probabilistic stay-time distribution, which generates the specific stay time of the visited-POIs. For example, it takes a short time to buy a beverage at a store and a long time to have dinner at a restaurant. We calculate the likelihood of a given stay-point and a visited-POI candidate using the probabilistic density function of the stay-time distribution. We adopt log-normal distribution as the stay-time distribution since it is a common choice when modeling the distribution of time spent in user activities [14, 28]. We use maximum likelihood estimation (MLE) to calculate the distribution parameters from the annotated data.

The MLE values for the log-normal distribution parameters of category c are calculated by

$$\nu_c = \frac{1}{|I_c|} \sum_{i \in I_c} \ln(sp_i.st), \quad \tau_c = \frac{1}{|I_c|} \sum_{i \in I_c} \left\{ \ln(sp_i.st) - \tau_c \right\}^2.$$

Here  $I_c \equiv \{i | i \in I, \operatorname{cat}(sp_i.poi) = c\}$  denotes the index set of the stay-points with corresponding visited-POIs whose category is c.  $|I_c|$  denotes the size of set  $I_c$ , which is the total number of visits to POI-category c. Using these parameters, we calculate feature value  $x_3^{(v_{ik})}$ :

$$x_3^{(v_{ik})} = \mathcal{LN}(sp_i.st; \nu_{\operatorname{cat}(k)}, \tau_{\operatorname{cat}(k)}),$$

where  $\mathcal{LN}$  is the probability density function of the log-normal distribution.

#### 4.4.3 POI-POI INTERACTION FEATURES

The third part of the objective function calculates the validity in terms of the visited-POI combinations. This factor helps model the consistency of the visited-POIs in a session because all of the pairs of the visited-POI candidates are considered. This aspect is also a key difference between our method and the conventional methods. In this paper, we prepared two POI-POI features: (1) a probability of POI-POI category dependency and (2) a Jaccard coefficient between two POI categories.

The first feature,  $x_1^{(t_{ijkl})}$ , is calculated using the probability of two POI categories:

$$x_1^{(t_{ijkl})} = P(\operatorname{cat}(l)|\operatorname{cat}(k)) = \frac{\operatorname{num}(\operatorname{cat}(k) \to \operatorname{cat}(l)) + \beta}{\operatorname{num}(\operatorname{cat}(k)) + \beta|C|},$$

where num(cat(k)) denotes the number of visits to cat(k) in the annotated data, num(cat(k)  $\rightarrow$  cat(l)) denotes the number of visits to cat(l) after a visit to cat(k),  $\beta$  denotes the smoothing parameter, and |C| denotes the total number of categories.

As a second feature, we use a Jaccard coefficient, a well-known technique in data mining, to capture the connection between two factors. We calculate the Jaccard coefficient between two POI categories by

$$x_2^{(t_{ijkl})} = \operatorname{Jaccard}(\operatorname{cat}(k), \operatorname{cat}(l)) = \frac{|S(\operatorname{cat}(k) \cap \operatorname{cat}(l))|}{|S(\operatorname{cat}(k))| + |S(\operatorname{cat}(l))|}$$

where  $S(\operatorname{cat}(k) \cap \operatorname{cat}(l))$  denotes a set of sessions containing both categories  $\operatorname{cat}(k)$  and  $\operatorname{cat}(l)$ .

#### 4.4.4 Sequence length features

This part plays the role of a regularizer that controls the number of visited-POIs in a session. That is, it enables the objective function to avoid assigning too many or too few visited-POIs. We prepared binary variable  $y_m$  that takes 1 if the number of visited-POIs is m and 0 otherwise.

We assume that the validity of the visited-POI number can be modeled by Gaussian distribution and use its probability density function to create feature  $x_1^{(y_m)}$ :  $x_1^{(y_m)} = \mathcal{N}(m; \mu_y, \sigma_y^2)$ , where  $\mu_y$  and  $\sigma_y^2$  are the mean and variance parameters. Each user has his/her own such parameters, which we estimate on the basis of MLE from the user's annotated data.

## 4.4.5 Constraints

Each constraint in problem P plays the important role of satisfying the requirements for the visited-POI assignment task. For example,  $s_i = 1$  and  $v_{ik} = 0 \forall k$  are supposedly impermissible results. In this example,  $s_i = 1$  indicates that stay-point i is a significant location, and  $v_{ik} = 0 \forall k$  indicates there is no visited-POI in stay-point i. These results contradict our assumption. To avoid such problems, we introduce our assumptions into the problem as constraints and prepare the following:

- Eq. (10): The left constraint ensures that stay-point i is either a significant location or a non-significant location. The right constraint ensures the exclusive selection of overlapped stay-points.
- Eq. (11): If stay-point i is a significant location, there is one corresponding visited-POI.
- Eq. (12): There is only one dependency pattern from stay-points i to j.
- Eq. (13): If  $t_{ijkl} = 1$ , visited-POI at stay-point *i* is POI *k*, and visited-POI at stay-point *j* is POI *l*.
- Eq. (14): A session length, which is exactly a single value (left constraint), is calculated by the sum of the adopted significant locations (right constraint).

Note that the exclusive selection factor in Eq. (10) plays a particularly important role when different algorithms are used to extract stay-points because some cannot be adopted as significant locations.

## 4.5 Algorithm

We show our algorithm in Fig. 4. It receives a set of stay-points in a session as given significant location candidates and weight parameters. For each significant location candidate, its visited-POI candidates are obtained by a POI database  $\mathcal{P}_{p+c}$ . It also creates features for each stay-point and a corresponding visited-POI candidate and generates an ILP problem on the basis of the calculated feature and the given weight parameters. It then solves the problem using a general ILP solver. Finally, the algorithm returns significant locations and their visited-POIs **V** as results.

Algorithm Joint Estimation of Significant Location
and its Visited-POI
<b>Input</b> : $SP = (sp_i)_{i=1}^N, \mathbf{w}^{(s)}, \mathbf{w}^{(\bar{s})}, \mathbf{w}^{(v)}, \mathbf{w}^{(t)}, \mathbf{w}^{(y)}$
Output: V
1: FOR $i$ in 1 to $N$
2: Create features $\mathbf{x}^{(s_i)}$ and $\mathbf{x}^{(\bar{s}_i)}$ .
3: Obtain POI candidates $\{poi_k\}_{k=1}^{K_i}$ .
4: <b>FOR</b> $k$ in 1 to $K_i$
5: Create feature $\mathbf{x}^{(v_{ik})}$ .
6: ENDFOR
7: ENDFOR
8: Create feature $\mathbf{x}^{(t_{ijkl})}$ .
9: Create feature $\mathbf{x}^{(y_m)}$ .
10: Generate 0-1 ILP code and solve the problem.
11: Read $\mathbf{V}$ from the solution.
12: <b>RETURN V</b>

Figure 4: Algorithm of our framework

## 4.6 Computational complexity

It is computationally expensive to solve the personalized visited-POI estimation task with incorporating all the long-distance dependencies by a naive exhaustive search since we need to consider all the combinatorial dependencies among all the estimated significant locations and their visited-POIs. However, a recent-advanced general purpose ILP solver (e.g., Gurobi, and CPLEX) can solve many ILP problems in a practical time even if a given problem is NP-hard since it equips various strong techniques for efficiently solve sophisticated combinatorial problems. In fact, the number of variables in our problem is relatively very small in terms of a standard problem size handled currently in generic ILP solvers, and thus, the execution time for estimating visited-POIs from a single session (one day's data) by our method generally takes less than one second (see details in experimental results).

Note that we can model long-distance dependencies by the k-th order Markov model with a sufficiently large k. Theoretically, its calculation cost is a polynomial time if we apply an appropriate dynamic programming algorithm. However, it actually becomes a large k-th polynomial time, for example k = 20, which is basically infeasible.

# 5. Evaluation

# 5.1 Data

We collected the actual GPS trajectories and the corresponding visited-POI annotations of users to evaluate our proposed task: personalized visited-POI assignments<sup>4</sup>. First, we collected the trajectories of three subjects (users) in the first round of our challenge for

<sup>4.</sup> Unfortunately, no publicly available dataset contains both individual GPS trajectories and user's visitedlocation information.

Table 2: Statistics for annotated dataset. (#Sessions: number of sessions. #POIs: number of POIs. Ave. POI/Session: average number of POIs per session. #Uniq. POIs: number of unique POIs per user. #count=1 POIs: number of POIs appeared only once per user. #Uniq. POI cate.: number of unique POI categories.)

User	#Sessions	#POIs	(Ave. POI/Session)	#Uniq. POIs	(#count=1 POIs)	#Uniq. POI cate.
A01	16	98	(6.13)	24	(15)	18
A02	20	89	(4.45)	43	(31)	28
A03	18	95	(5.28)	43	(35)	32
B04	21	119	(5.67)	56	(46)	32
B05	16	112	(7.00)	41	(31)	26
B06	18	116	(6.44)	60	(50)	37
B07	19	99	(5.21)	47	(40)	27
B08	19	115	(6.05)	35	(26)	19
B09	20	221	(11.05)	119	(97)	36
B10	19	120	(6.32)	41	(26)	21
B11	20	168	(8.40)	92	(77)	39
B12	17	162	(9.53)	43	(32)	21
B13	20	159	(7.95)	84	(62)	52
B14	10	67	(6.70)	38	(33)	30
B15	17	87	(5.12)	39	(33)	28
B16	21	85	(4.05)	16	(11)	13
B17	19	85	(4.47)	30	(21)	18
B18	19	87	(4.58)	41	(37)	20
B19	21	95	(4.52)	39	(32)	20
B20	16	74	(4.63)	40	(38)	31
B21	21	88	(4.19)	20	(11)	11
ALL	387	2341	(6.05)	971	(758)	202

preliminary evaluations and obtained 18 more from the second round. Thus, we collected a total of 21 subject trajectories and corresponding visited-POI annotations. Each subject carried a mobile device with a logging application for three weeks (21 days) for both the first and second rounds <sup>5</sup>. The logging application stores position information every three seconds with an Android OS location class that uses both GPS and WiFi positioning systems. The application automatically adopts the position with a higher accuracy value. Subjects also assigned visited-POIs to their significant locations. The visited-POI candidates were collected by Foursquare Search Venues API<sup>6</sup>. If a true visited-POI did not appear in the collected POI candidates, then the subject added custom POIs. Table 2 shows the fundamental statistics of our dataset. The numbers from 01 to 21 in the first column indicate the

<sup>5.</sup> In Table 2, some subject trajectories were actually less than 21 sessions (days) for several reasons, e.g., they didn't go anywhere or didn't visit any significant locations (mostly on weekends), forgot to turn on (or bring) the system, or privacy issues.

<sup>6.</sup> https://developer.foursquare.com/docs/venues/search

User	Avg. distance [m]	(longest)	Ave. rank	#Top-rank	(ratio)	#Worst-rank
A01	$32.57 \pm 15.49$	(79.56)	$5.74 \pm 8.71$	36 / 98	(0.3673)	48
A02	$37.00 \pm 60.08$	(298.88)	$12.80 \pm 24.40$	40 / 89	(0.4494)	130
A03	$28.78\pm37.35$	(332.61)	$12.70 \pm 20.57$	19 / 95	(0.2000)	111
B04	$34.30 \pm 39.89$	(242.91)	$13.01 \pm 14.58$	11 /119	(0.0924)	67
B05	$49.04 \pm 40.74$	(171.55)	$18.17 \pm 21.39$	11 / 112	(0.0982)	66
B06	$49.37 \pm 49.34$	(290.92)	$16.07 \pm 19.85$	22/116	(0.1897)	77
B07	$33.72 \pm 40.09$	(199.38)	$9.25 \pm 12.35$	17 / 99	(0.1717)	48
B08	$37.67 \pm 30.80$	(149.07)	$9.46 \pm 11.68$	14 / 115	(0.1217)	61
B09	$28.70 \pm 39.08$	(209.27)	$5.05 \pm 8.76$	73/221	(0.3303)	58
B10	$40.00 \pm 35.42$	(196.41)	$11.56 \pm 14.20$	5/120	(0.0417)	71
B11	$39.04 \pm 46.33$	(320.95)	$10.45 \pm 15.16$	57/168	(0.3393)	73
B12	$47.46 \pm 34.79$	(187.65)	$15.68 \pm 14.72$	9/162	(0.0556)	68
B13	$50.31 \pm 72.54$	(475.26)	$16.34 \pm 17.03$	18/159	(0.1132)	60
B14	$26.93 \pm 46.07$	(174.84)	$10.05 \pm 18.24$	12 / 67	(0.1791)	68
B15	$29.80 \pm 32.07$	(151.67)	$8.68 \pm 10.92$	12 / 87	(0.1379)	60
B16	$108.67 \pm 131.1$	(465.27)	$3.66 \pm 7.06$	23 / 85	(0.2706)	36
B17	$33.22 \pm 25.15$	(102.37)	$13.14 \pm 15.75$	11 / 85	(0.1294)	72
B18	$42.22 \pm 32.75$	(125.19)	$16.60 \pm 16.54$	4 / 87	(0.0460)	67
B19	$44.11 \pm 68.88$	(380.89)	$17.65 \pm 17.89$	5 / 95	(0.0526)	61
B20	$26.63 \pm 20.30$	(67.26)	$12.89 \pm 13.76$	5 / 74	(0.0676)	47
B21	$31.77 \pm 22.25$	(103.79)	$13.71 \pm 14.15$	6 / 88	(0.0682)	48
ALL	$39.66 \pm 49.39$	(475.26)	$15.40\pm21.20$	410 / 2341	(0.1751)	130

Table 3: Statistics of our data: distance from stay-point center to visited-POI and average ranking of true visited-POIs among visited-POI candidates based on distance

subject IDs. Single letters 'A' and 'B' of the prefixes of the subject IDs represent one of two distinct time periods during which the data were actually collected (A for the first round and B for the second). We collected almost 400 sessions and over 2,300 actual visited-POIs, which is a relatively large amount of GPS data with actual user visiting histories.

## 5.2 Preliminary data analyses

----

In this section, we describe our preliminary data analyses that were explained in the previous section.

#### 5.2.1 Analysis of visited-POIs:

We obtained the visited-POI candidates near the center of the significant locations in the dataset using the Foursquare Search Venues API. We calculated the distance from the center of a significant location to the position of each visited-POI candidate as well as the distance's mean and standard deviation. After sorting the POI candidates by distance in

ascending order, we calculated the average rank of the visited-POIs and the ratio of the number of top-ranked POIs to the number of visited-POIs. We show the results in Table 3.

We confirmed that on average, the visited-POIs are mostly 100 meters or less from the center of the stay-point, and the average rank of the visited-POIs was 15.40. These statistics support that visited-POIs mostly exist near the center of the stay-points.

Next we discuss why a visited-POI rarely appears as the nearest neighbor of a staypoint center. We consider two reasons: (1) incorrect positioning of the stay-point center due to GPS/WiFi positioning errors and (2) the high density of POI candidates around the stay-points. The occurrence of positioning errors on GPS loggers is a well-known problem, especially for indoor environments [46]. Kjærgaard et al. [15] reported that even state-ofthe-art GPS loggers suffer from positioning errors caused by the environment. They found that less than 80% of the track-points in their experiments were within 20 meters of the true points. We conducted a preliminary experiment that identified that similar errors occurred for the track-points obtained outside a building. GPS positioning errors seem inevitable. As for the second reason (2), our study showed that each significant location has an average of 70.3 visited-POI candidates within 500 meters. This indicates that the existence of other POIs prevents a visited-POI from being the nearest neighbor. Thus, we have to predict visited-POIs from among all the POI candidates. This complicates solving the visited-POI assignment task with a simple combination of stay-point extraction and a POI recommendation technique.

#### 5.2.2 Analysis of stay-point extraction

We investigated the functionality of one of the conventional stay-point extraction methods to verify the validity of our hypothesis in Section 4. We used this method because staypoint extraction is generally the first step for solving GPS trajectory mining tasks. We also sought to confirm the effectiveness of the stay-point extraction approach if it were used to perform the first step in our visited-POI assignment task.

For this experiment, we selected the method developed by Kang et al. [12] since it is one of the most widely used stay-point extraction methods [44, 45]. It has two different types of threshold parameters. One is a time threshold, which we refer to as  $\theta_{\text{time}}$ . The other is a distance threshold, which we refer to as  $\theta_{\text{dist}}$ . In most cases, these are preliminarily configured by hand. Kang's method recognizes continuous track-points as stay-points if the continuous track-points for all the positions are within a  $\theta_{\text{dist}}$  meter radius over  $\theta_{\text{time}}$  seconds. As an example of this method's usage, Zheng et al. used it to configure  $\theta_{\text{time}} = 1800$ ,  $\theta_{\text{dist}} = 200$  to extract stay-points [44].

We evaluated how Kang's method with various parameter settings performed the first step of the visited-POI assignment. If the algorithm successfully extracted a stay-point that includes the timestamp of an annotated visited-POI, we evaluated it as correct and refer to it as a positive  $(TP_{sp})$ . We evaluated the extracted stay-point as a false positive  $(FP_{sp})$  if it does not include any annotated true visited-POIs. On the other hand, we evaluated an annotated true visited-POI as a false negative  $(FN_{sp})$  if the algorithm failed to extract any stay-point that includes it. We used  $TP_{sp}$ ,  $FP_{sp}$ , and  $FN_{sp}$  to calculate precision  $PRE_{sp}$ 

$\theta_{\rm dist}$	$ heta_{ ext{time}}$	$PRE_{sp}$	$REC_{sp}$	$TP_{sp}$	$FP_{sp}$	$FN_{sp}$
100	1800	0.857	0.367	36	6	62
100	900	0.854	0.418	41	7	57
100	180	0.541	0.867	85	72	<u>13</u>
200	1800	0.889	0.327	<u>32</u>	$\underline{4}$	66
200	900	0.870	0.408	40	6	58
200	180	0.447	0.735	72	89	26
500	1800	0.889	0.327	<u>32</u>	<u>4</u>	66
500	900	0.811	0.439	43	10	55
500	180	0.380	0.612	60	98	38

Table 4: Evaluation of Kang's stay-point extraction algorithm using various parameter combinations on user A01's dataset. Bold font numbers indicate maximum values, and underlined numbers indicate minimum values in columns.

and recall  $REC_{sp}$  of stay-point extraction with the following equation:

$$PRE_{sp} = \frac{TP_{sp}}{TP_{sp} + FP_{sp}}, \quad REC_{sp} = \frac{TP_{sp}}{TP_{sp} + FN_{sp}}.$$
(16)

Note that there is no chance a true visited-POI is assigned if the algorithm failed to extract any stay-point that included it. Therefore, to achieve highly accurate visited-POI assignments, the stay-point extraction algorithm must identify the stay-points as near to the true visited-POIs as possible.

Table 4 shows the number of stay-points correctly and erroneously extracted from GPS trajectories under various threshold settings. It also shows the stay-point extraction precision and recall values obtained using Kang's method with combinations of  $\theta_{\text{dist}} \in \{100, 200, 500\}$  and  $\theta_{\text{time}} \in \{180, 900, 1800\}$ . We determined that an extracted stay-point was correct if it were less than 50 meters from its nearest true significant location. In the table, bold font numbers denote maximum values and the underlined numbers denote minimum values in the columns. We also analyzed the stay-time distribution of the visited-POIs in the dataset. Fig. 5 shows the frequency and the cumulative ratios of the stay time.

Table 4 clearly shows that with  $\theta_{\text{dist}} = 200$ ,  $\theta_{\text{time}} = 1800$  parameters, Kang's method achieved a high precision of 0.889 but a low recall of 0.327. From Fig. 5, over 67% of the stay-points had a stay time less than 30 minutes. This indicates that with the above settings Kang's method failed to extract more than 67% of the stay-points. Therefore, with these settings, high recall was not achieved in our task. On the other hand, with  $\theta_{\text{dist}} = 100$ ,  $\theta_{\text{time}} = 180$  parameters, Kang's method achieved a high recall of 0.867 while showing the lowest FN number in the column. Although high recall performance was achieved with these settings, the precision value was low. The most critical observation is that most true significant locations can be obtained using a conservative threshold parameter setting (e.g.,  $\theta_{\text{dist}} = 100$ ,  $\theta_{\text{time}} = 180$ ), even though the extracted stay-points contain many false positives.

Note that the recall values in Table 4 are the upper bound recall values in the visited-POI assignment tasks. The algorithm cannot estimate the visited-POIs if there are no extracted



Figure 5: Stay-time distribution of true significant locations in all users annotated dataset. Bars denote frequency and line denotes cumulative ratio.

stay-points  $(FN_{sp})$ . Similarly, the algorithm always fails to estimate the visited-POIs if the extracted stay-points are  $(FP_{sp})$ . Thus, stay-point extraction errors directly lowered the accuracy of the visited-POI assignments. This supports the validity of our proposed method's design, which first allows  $FP_{sp}$  to exhaustively extract stay-point candidates and then simultaneously estimates the validity of the candidates and their visited-POIs to determine the best combination of stay-points and visited-POIs as sequences.

## 5.3 Evaluation Setting

#### 5.3.1 Comparative Methods:

We compared the following methods to verify the effectiveness of the joint estimation method.

- NN: The first is the Nearest Neighbor (NN) method, which selects the nearest neighbor POI from a stay-point center. NN resembles the visited-POI version of Cao et al. [5].
- 2. **NCI**: The second is a popularity-based method, which selects the most popular POI in terms of the number of check-ins (NCI) from neighbor POIs.
- 3. Rank(LI11): The third is a supervised learning-to-rank based method developed by Lian et al. [20]. We used RankLib<sup>7</sup> as an implementation of learning-to-rank algorithms.
- 4. **Rank(SH13)**: The fourth is another supervised learning-to-rank based method developed by Shaw et al. [32]<sup>8</sup>. We also used RankLib and Rank(LI11).

<sup>7.</sup> http://sourceforge.net/p/lemur/wiki/RankLib/

<sup>8.</sup> We removed from our implementation some of the features used by Shaw et al. [32] in their method because we could not obtain them from Foursquare API.

In addition, the method proposed in Lv et al. [27] uses a stay-point extraction algorithm and a semantic location extraction algorithm in a cascaded manner as well as our approach. The learning framework of their method resembles a special case of the ranking approach for Rank(LI11) and Rank(SH13) since they did not handle any POI-POI dependency information. Moreover, their approach basically detects frequently visited places, which is not suitable for our task. Therefore, we do not directly compare their method in our experiments and assume that in our task Rank(LI11) and Rank(SH13) are alternatives for Lv et al.

- 5. **JE**: The fifth is the joint estimation method that considers the higher-order relation between every interaction among the information of significant locations. JE, which represents the full version of the proposed method in this paper, is our primary method.
- 6. CRF(JE<sub>1od</sub>): The sixth is first-order CRF (or linear-chain CRF). CRF(JE<sub>1od</sub>) resembles (a modified version of) the method proposed in Wang et al. [34] that is adopted to fit the visited-POI assignment task. Note that CRF(JE<sub>1od</sub>) remains part of our method since it can be interpreted as a degraded version of JE. The difference between JE and CRF(JE<sub>1od</sub>) is how they calculate the POI-POI features. CRF(JE<sub>1od</sub>) only incorporates the POI-POI features obtained from neighbors in a session.

These six methods all use the same extracted stay-points as candidates of significant locations.

#### 5.3.2 EXPERIMENTAL SETTINGS:

We used Kang's method as a stay-point extraction algorithm. As we confirmed in the previous section, our framework assumes exhaustively extracted significant location candidates as input. Therefore, we used a parameter set of ( $\theta_{\text{dist}} = 100$ ,  $\theta_{\text{time}} = 180$ ).

Basically, we split individual data into two parts, training and test. For Rank(LI11) and Rank(SH13), the training data were used to build a ranking model. For JE(full) and CRF(JE<sub>1od</sub>), we used training data to create features and select weight parameter  $\mathbf{w} \in \{0.1, 1.0\}^W$  with the highest F1-score by a grid-search, where  $\mathbf{w} = (\mathbf{w}^{(s)}, \mathbf{w}^{(\bar{s})}, \mathbf{w}^{(v)}, \mathbf{w}^{(t)}, \mathbf{w}^{(y)})$  and W denotes the dimension number of  $\mathbf{w}$ . We used Gurobi Optimizer v.7.5<sup>9</sup> as an ILP solver for JE(full) and CRF(JE<sub>1od</sub>). We set  $\gamma = 0.1$ ,  $\lambda = 1/30$ ,  $\alpha_1, \alpha_2 = 0.9$ ,  $\beta = 0.01$  for feature calculation<sup>10</sup>. Then we used ten-fold cross-validation as the evaluation setting to test the fundamental performance of the visited-POI assignment on several comparative methods.

## 5.3.3 Evaluation measure:

All the experiments in this paper were evaluated based on their F1-score, calculated by the harmonic mean of the precision and recall scores. Fig. 6 shows how the F1-score is measured from the true significant locations and their visited-POIs. Our dataset contains the true

<sup>9.</sup> http://www.gurobi.com/

<sup>10.</sup> We selected these parameters based on the results of preliminary experiments on the data obtained in the first round (A01, A02, and A03).



Figure 6: Evaluation method for visited-POI assignment task

significant locations and the corresponding visited-POIs assigned by the users. In the figure, there are four significant locations and four visited-POIs as ground-truth labels. A method selects three stay-points and assigns a visited-POI to them in this example. First, we match a selected stay-point to the true significant location that contains the middle timestamp  $((sp_i.bt + sp_i.et)/2)$  of the *i*-th stay-point. Two of the three selected stay-points match the true significant locations in the example. If an extracted stay-point matches the true significant location, we can determine whether the predicted visited-POI agrees with the true visited-POI assignment. The prediction falls under true positive  $(TP_{POI})$  if it agrees with the ground-truth label and falls under false positive  $(FP_{POI})$  otherwise. We always judge the selected stay-points that do not match the true significant locations as  $FP_{POI}$ . We also assess a true significant location that is not matched by any selected stay-point as a false negative  $(FN_{POI})$ .

With  $TP_{POI}$ ,  $FP_{POI}$  and  $FN_{POI}$ , we can calculate the precision  $(PRE_{POI})$  and recall  $(REC_{POI})$  scores in a similar way as in Eq. (16), namely;

$$PRE_{\rm POI} = \frac{TP_{\rm POI}}{TP_{\rm POI} + FP_{\rm POI}}, \quad REC_{\rm POI} = \frac{TP_{\rm POI}}{TP_{\rm POI} + FN_{\rm POI}}.$$
 (17)

Then F1-score (F1) is obtained by its harmonic mean;

$$F1_{POI} = \frac{2PRE_{POI}REC_{POI}}{PRE_{POI} + REC_{POI}} = \frac{2TP_{POI}}{2TP_{POI} + FN_{POI} + FP_{POI}}.$$
 (18)

#### 5.4 Results

#### 5.4.1 Performance comparison among comparative methods:

Fig. 7 shows the ten-fold cross-validation results for each individual user. The x-axis is each user ID, where Micro and Macro respectively represent the micro and macro averages of all the user results.

At first, we observed that JE(full) obtained the best F1 scores on 17 out of 21 users. We conducted Wilcoxon signed rank test at a confidence level of 0.01 on the Macro averages between all the pairs of JE(full) and other comparative methods, namely (1) JE(full) vs. NN, (2) JE(full) vs. NCI, (3) JE(full) vs. Rank(LI11), (4) JE(full) vs. Rank(SH13), and



Figure 7: F1-scores on ten-fold cross-validation

(5) JE(full) vs.  $CRF(JE_{1od})$ . We confirmed that there were statistical differences on all the five significance tests. Therefore, we conclude that JE(full) significantly outperformed the other comparative methods.

Next, NN, NCI, Rank(LI11), and Rank(SH13) are point-wise estimation approaches, whereas  $CRF(JE_{1od})$  and JE(full) are joint estimation approaches. In our evaluation, the joint estimation approach is preferable for visited-POI assignments. This implies that visited-POIs are collected each other.

Moreover, JE(full) outperformed  $CRF(JE_{1od})$  in the same joint estimation approach. This fact indicates that long-distance dependency information is essential for our task since the difference between JE(full) and  $CRF(JE_{1od})$  addresses whether the methods incorporate long-distance transition information. This explains the need to formulate the task as an ILP problem for easily and efficiently incorporating the information.

## 5.4.2 Effectiveness of individual visited-POI assignment setting

We conducted additional experiments to verify the idea that personal annotation, rather than annotations made by all users, contains essential information to achieve high accuracy in our task. Fig. 8 shows the result of each comparative method when we mixed all the user data. An additional label, Mix, represents the results in the *mixed setting*; we trained the models using all the user training data all at once, and evaluated each user's test data by the trained model. Another additional label, Pers, represents the results in the *personalized setting*. These results are the macro-averages of the F1-scores of the individual visited-POI assignments for each method in Fig. 7.



Figure 8: F1-scores on mixed and personalized settings for each method

At first, the NN and NCI results are identical since these methods do not perform any training using the training data. Therefore, the results are always equivalent regardless of the training data. Then, clearly, the Pers results were consistently better than Mix results. We also confirmed that results of the Pers setting on JE(full),  $CRF(JE_{1od})$  and Rank(LI11) have significant difference on Wilcoxon signed rank test at a confidence level of 0.01 from those of the Mix setting. Note here that the training data sizes of the personalized setting were approximately 21 times smaller on average than the mixed setting. However, the personalized setting consistently outperformed the mixed setting. These are surprising results. This observation is empirical evidence for the effectiveness of personalized visited-POI assignments.

#### 5.5 Efficiency analysis

We investigated the actual elapsed time for JE(full) and the other comparative methods since solving ILP problems entails high computational cost. We measured the average elapsed time for splitting all of the user data into numbers of stay-points. Fig. 9 shows the results, where the x-axis denotes the number of stay-points in the session and the y-axis denotes the actual elapsed time [sec.]. The experiment was conducted on a single machine.

Our joint estimation approach, JE(full), clearly took more calculation time than the other comparative methods. This observation is expected since JE(full) runs an ILP solver to obtain its result, but the other methods (NN, NCI, and Rank) estimate the results in a point-wise estimation manner. Therefore, there is a trade-off between calculation cost and performance.



Figure 9: Elapsed time over number of stay-points for each comparative method

However, here the elapsed time of JE(full) actually averaged less than 0.4 seconds, suggesting that JE(full) can be run in a practical time. This is because we only need to run the system once a day in our setting: one second per day for a single user. Although the current performance of smartphones/tablets is much inferior to that of the machine used in our experiment, the experimental settings remain realistic since smartphones/tablets need a few seconds for processing a single user's private data. We believe our results confirm the efficiency of our algorithm.

## 5.6 Evaluation of sequential visited-POI assignments

A cross-validation evaluation may not follow the actual use-case of visited-POI assignment tasks. Therefore, we conducted additional experiments to verify the effectiveness of actual usage. To resemble the actual usage of the visited-POI assignment, we considered the following procedure as an evaluation setting:

- 1. Train a personalized model for each user using the first three sessions (days) as training data.
- 2. Evaluate the next days' session (the fourth session for first evaluation round, and always evaluate a single session in an evaluation).
- 3. Re-train (or update) the model by adding the evaluated data used in (2) to the training data.
- 4. Repeat (2) and (3) alternatively until the last session is evaluated in the procedure (2).
- 5. Report the average performance of the entire evaluation in the procedure (2).



Figure 10: Brief sketch of procedure of our sequential evaluation

The above process resembles a daily update system that estimates every session once a day and then updates the model. We refer to this configuration as a sequential evaluation. Fig. 10 shows a brief sketch of our sequential evaluation procedure.

Fig. 11 shows the results on a sequential evaluation setting. Compared with the results in Fig. 7, the Micro and Macro average worsened in all the experiments. This is because the sequential evaluation's task setting is more difficult than the cross-validation since the amount of training data becomes much smaller at the sequential evaluation's beginning. However, the tendency of the performance gaps among comparative methods resembles those observed in Fig. 11. We observed that JE(full) still obtained the best F1 scores on 16 out of 21 users. We also conducted Wilcoxon signed rank test at a confidence level of 0.01 on the Macro averages between all the pairs of JE(full) and other comparative methods, as described in Section 5.4.1, namely (1) JE(full) vs. NN, (2) JE(full) vs. NCI, (3) JE(full) vs. Rank(LI11), (4) JE(full) vs. Rank(SH13), and (5) JE(full) vs. CRF( $JE_{1od}$ ). We also confirmed that there were statistical differences on all the five significance tests.



Figure 11: Results on sequential evaluation

Thus, JE(full) significantly outperformed the other comparative methods on the setting of sequential visited-POI assignments.

# 6. Conclusion

We tackled the problems involved in a personalized visited-POI assignment task that assigned visited-POIs to their corresponding significant locations in given individual GPS trajectory data with partially annotated user data and a POI database. We developed a novel visited-POI selection framework based on 0-1 ILP formulation that selects true significant locations and simultaneously assigns visited-POIs while considering different aspects of the selected significant locations and assigned visited-POIs. Experimental results showed that a conventional stay-point extraction algorithm cannot simultaneously achieve both precision and recall when extracting true significant locations. Our results also showed that, for performing the visited-POI assignment task, the framework we developed outperforms conventional methods using various cascaded procedures. Although our method solves an ILP problem that entails high computational cost, we confirmed that it can also output results in a practical time.

# References

 B. Adams, D. Phung, and S. Venkatesh. Extraction of social context and application to personal multimedia exploration. In *Proc. ACM MM '06*, pages 987–996, 2006.

- [2] D. Ashbrook. Learning significant locations and predicting user movement with GPS. In Proc. ISWC '02, pages 101–108, 2002.
- [3] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7(5):275–286, 2003.
- [4] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proc. SIGSPATIAL '12*, pages 199–208, 2012.
- [5] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from GPS data. Proc. VLDB Endow., 3(1-2):1009–1020, 2010.
- [6] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proc. ICML* '07, pages 129–136.
- [7] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.
- [8] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In Proc. KDD '11, pages 1082–1090.
- [9] Lisette Espín Noboa, Florian Lemmerich, Philipp Singer, and Markus Strohmaier. Discovering and characterizing mobility patterns in urban spaces: A study of manhattan taxi data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 537–542, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.
- [10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD* '96, pages 226–231, 1996.
- [11] K. Farrahi and D. Gatica-Perez. Discovering routines from large-scale human locations using probabilistic topic models. ACM Trans. Intell. Syst. Technol., 2(1):3:1–3:27, 2011.
- [12] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. In *Proc. WMASH* '04, pages 110–118, 2004.
- [13] Ilkcan Keles, Matthias Schubert, Peer Kröger, Simonas Šaltenis, and Christian S. Jensen. Extracting visited points of interest from vehicle trajectories. In Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data, GeoRich '17, pages 2:1–2:6, New York, NY, USA, 2017. ACM.
- [14] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In Proc. INFOCOM '06, 2006.
- [15] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk. Indoor positioning using gps revisited. In *Proc. Pervasive '10*, pages 38–56, 2010.

- [16] J. Krumm and E. Horvitz. Predestination: Where do you want to go today? Computer, 40(4):105–107, 2007.
- [17] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *Proc. CIKM '10*, pages 579–588, 2010.
- [18] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. CLR: A collaborative location recommendation framework based on co-clustering. In Proc. SIGIR '11, pages 305–314, 2011.
- [19] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma. Mining user similarity based on location history. In Proc. GIS '08, pages 34:1–34:10, 2008.
- [20] D. Lian and X. Xie. Learning location naming from user check-in histories. In Proc. GIS '11, pages 112–121, 2011.
- [21] D. Lian and X. Xie. Mining check-in history for personalized location naming. ACM Trans. Intell. Syst. Technol., 5(2):32:1–32:25, 2014.
- [22] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In Proc. IJCAI '05, pages 773–778.
- [23] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. Int. J. Rob. Res., 26(1):119–134, 2007.
- [24] J. Liu, O. Wolfson, and H. Yin. Extracting semantic location from outdoor positioning systems. In Proc. MDM '06, pages 73–73, 2006.
- [25] Tie-Yan Liu. Learning to Rank for Information Retrieval. Springer, 2011.
- [26] Xinjiang Lu, Zhiwen Yu, Leilei Sun, Chuanren Liu, Hui Xiong, and Chu Guan. Characterizing the life cycle of point of interests using human mobility patterns. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '16, pages 1052–1063, New York, NY, USA, 2016. ACM.
- [27] Mingqi Lv, Ling Chen, and Gencai Chen. Discovering personally semantic places from gps trajectories. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, pages 1552–1556, New York, NY, USA, 2012. ACM.
- [28] D. Minder, P. J. Marrón, A. Lachenmann, and K. Rothermel. Experimental construction of a meeting model for smart office environments. In Proc. REALWSN '05, 2005.
- [29] K. P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.
- [30] Kyosuke Nishida, Hiroyuki Toda, and Yoshimasa Koike. Extracting Arbitrary-shaped Stay Regions from Geospatial Trajectories with Outliers and Missing Points. In IWCTS Proceedings of the th ACM SIGSPATIAL International Workshop on Computational Transportation Science, pages 1–7, October 2015.
- [31] D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In Proc. ICML '05, pages 736–743, 2005.

- [32] B. Shaw, J. Shea, S. Sinha, and A. Hogue. Learning to rank for spatiotemporal search. In *Proc. WSDM '13*, pages 717–726, 2013.
- [33] H. Wang, M. Terrovitis, and N. Mamoulis. Location recommendation in location-based social networks using user check-in data. In *Proc. SIGSPATIAL '13*, pages 364–373, 2013.
- [34] Weiqing Wang, Hongzhi Yin, Shazia Sadiq, Ling Chen, Min Xie, and Xiaofang Zhou. SPORE: A sequential personalized spatial item recommender system. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pages 954–965. IEEE, 2016.
- [35] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. Finding similar users using category-based location history. In Proc. GIS '10, pages 442–445, 2010.
- [36] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proc. ICDE* '13, pages 254–265, 2013.
- [37] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proc. SIGIR* '11, pages 325–334, 2011.
- [38] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng. Semantic trajectory mining for location prediction. In Proc. GIS '11, pages 34–43, 2011.
- [39] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and pois. In Proc. KDD '12, pages 186–194, 2012.
- [40] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. Magnenat-Thalmann. Time-aware point-ofinterest recommendation. In Proc. SIGIR '13, pages 363–372, 2013.
- [41] Ali Zarezade, Sina Jafarzadeh, and Hamid R. Rabiee. Spatio-temporal modeling of check-ins in location-based social networks. CoRR, abs/1611.07710, 2016.
- [42] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma. Understanding transportation modes based on GPS data for web applications. ACM Trans. Web, 4(1):1:1–1:36, 2010.
- [43] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw GPS data for geographic applications on the web. In *Proc. WWW '08*, pages 247–256, 2008.
- [44] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma. Recommending friends and locations based on individual location history. ACM Trans. Web, 5(1):5:1–5:44, 2011.
- [45] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proc. WWW '09*, pages 791–800, 2009.
- [46] Y. Zheng and X. Zhou, editors. Computing with Spatial Trajectories. Springer, 2011.