Inferring Trip Occupancies in the Rise of Ride-Hailing Services

Meng-Fen Chiang[1], Ee-Peng Lim[1], Wang-Chien Lee[2], Tuan-Anh Hoang[3]
[1]Living Analytics Research Centre, Singapore Management University
[2]Department of Computer Science and Engineering, The Pennsylvania State University
[3]L3S Research Center, Leibniz University of Hanover
{mfchiang,eplim}@smu.edu.sg,wlee@cse.psu.edu,hoang@l3s.de

ABSTRACT

The knowledge of all occupied and unoccupied trips made by self-employed drivers are essential for optimized vehicle dispatch by ride-hailing services (e.g., Didi Dache, Uber, Lyft, Grab, etc.). However, vehicles' occupancy status is not always known to service operators due to adoption of multiple ride-hailing apps. In this paper, we propose a novel framework, Learning to INfer Trips (LINT), to infer occupancy of car trips by exploring characteristics of observed occupied trips. Two main research steps, *stop point classification* and *structural segmentation*, are included in LINT. In the first step, we represent a vehicle trajectory as a sequence of stop points, and assign stop points with *pick-up*, *drop-off*, and *intermediate* labels thus producing a stop point label sequence. In the second step, for structural segmentation, we further propose several segmentation algorithms, including *greedy segmentation (GS)*, *efficient greedy segmentation (EGS)*, and *dynamic programming-based segmentation (DP)* to infer occupied trip from stop point label sequences. Our comprehensive experiments on real vehicle trajectories from self-employed drivers show that (1) the proposed stop point classifier predicts stop point labels with high accuracy, and (2) the proposed segmentation algorithm GS delivers the best accuracy performance with efficient running time.

KEYWORDS

occupancy inference, trajectory segmentation, ride-hailing services

ACM Reference Format:
Meng-Fen Chiang[1], Ee-Peng Lim[1], Wang-Chien Lee[2], Tuan-Anh Hoang[3]. 2018. Inferring Trip Occupancies in the Rise of Ride-Hailing Services. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3269206.3272025

1 INTRODUCTION

Owing to the rise of sharing economy, ride-hailing services (e.g., Didi Dache, Uber, Grab, etc.) have become an integral part of public transport. For ride-hailing app operators, realtime vehicle *occupancy* status is an essential information as it offers opportunities to optimize vehicle dispatch. For example, an occupied vehicle should not be arranged for another booking before the current trip is completed. On the other hand, drivers without passengers should be alerted whenever there are new bookings nearby.

Often, ride-hailing apps collect vehicle trajectory data, whether the vehicles are in service or not. However, the occupancy status of vehicles is not always known for the following reason. Self-employed drivers tend to adopt multiple ride-hailing app(s) to receive bookings, making it challenging to gauge actual availability of each driver. For instance, suppose that one driver serves two ride-hailing app(s) *A* and *B*. When the driver takes a trip booked from *A*, the occupancy status of the driver is occupied from *A*'s perspective but remains unoccupied from *B*'s perspective because the trip is unknown to *B* and vice versa. Partial knowledge of booked trips can result in inaccurate estimation of drivers' availability, i.e., assuming a driver is still available to take passengers while in fact the driver is busy with another booked trip from other platforms. A straightforward solution for this is to unify the partial knowledge from all ride-hailing contenders. This is unfortunately infeasible in reality due to the competition among the contenders.

In this work, we consider the practical scenario where the vehicle movement trajectories of vehicles are collected by a single mobile ride-hailing app. Moreover, occupancy in some parts of the collected trajectory data, corresponding to booked trips, is known. From the overall vehicle trajectory and the observed occupied trips within the trajectory, we aim to derive the occupancy status for *unobserved trips* within the remaining trajectory. It is worth noting that the unobserved trips, while their occupancy is unobserved to the ride-hailing app, may actually contain occupied trips. Our empirical analysis shows that the unobserved trips dominate the entire collection of trajectories (e.g., 87.1% of all sampling points).

As illustrated in Figure 1(a), the observed/unobserved trips can be derived from two types of data: (1) vehicle trajectory and (2) booked trips in these trajectories. Accordingly, we formulate the *occupancy inference* problem as follows.

Problem 1.1. (Occupancy Inference) Given a dataset of vehicle trajectories and booked trips of participating vehicles, determine all unobserved occupied and unoccupied trips within the vehicle trajectories based on the booked trips.

Solving the occupancy inference problem would be more straightforward if we have both labeled occupied and unoccupied trips in the trajectory dataset. We can resort to supervised learning by identifying discriminative features to classify occupied and unoccupied trips [25][23]. For example, the authors in [25] proposed a supervised learning approach to infer the occupied/unoccupied status of taxi trajectories. They assume both occupied and unoccupied trips are observed in the input data, which may be valid for some taxicabs specially equipped to record occupancy status. Nevertheless, the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '18, October 22–26, 2018, Torino, Italy
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3272025

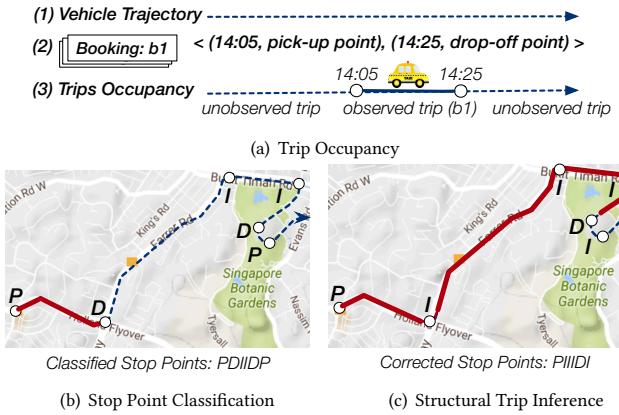
ride-hailing scenario considered in this paper is uniquely different. Due to the lack of detailed information on the unobserved trips, we do not have the ground truth for "unoccupied" trips but only some knowledge on a small subset of "occupied" trips (obtained from the booked trips). In other words, our prediction task is one with *single class* ground truth, which requires both new prediction methods and new accuracy evaluation framework.

To utilize observed occupied trips to infer unobserved occupied and unoccupied trips, we first determine the labels of some important trajectory points or stop points at which a vehicle has to make brief stops. We assume that each occupied trip consists of a pick-up point and a drop-off point, while having a number of intermediate stops due to traffic lights or other road conditions. Pick-up, drop-off, and intermediate *stop points* are all relatively stationary in a trajectory. Thus, we can represent a trajectory as a sequence of properly labelled stop points. Following the patterns of pick-up, intermediate stop points and drop-off in an occupied trip, we can segment a trajectory into occupied and unoccupied trips. Thus, in this work, we propose to first learn stop points from observed occupied trips. To label these stop points as pick-up, drop-off, or intermediate, we train stop point classifiers using features engineered from a set of labeled stop points, which can be extracted from known occupied trips. We then devise several algorithms to infer occupied trips by forming valid label sequences, i.e., an occupied trip is a sequence starting with a pick-up point, followed by a number of intermediate stop points before a drop-off stop point.

Figures 1(b) and 1(c) illustrate our proposed idea. Given a vehicle trajectory, we extract stop points (refer to Section 2 for stop point definition) based on a speed threshold. As shown in Figure 1(b), we obtain six stop points depicted as circles. We apply the stop point classifier to classify them into: *pick-up* points (P), *drop-off* points (D) and *intermediate* points (I). Suppose the sequence of assigned stop point labels is PDIIDP as shown in Figure 1(b). The first two label assignments PD is a well-formed occupied trip (red solid line) but not the remaining label assignments IIDP (which refers to neither an occupied nor an unoccupied trip). To refine the label sequence with least correction, we employ one of the proposed *Structural Trip Inference* algorithms to segment the trajectory into well-formed sequences (which represent either occupied or unoccupied trips). In the example, two corrections (2^{nd} and 6^{th} stop points) are made to transform the stop points in Figure 1(b) into the well-formed seqences in Figure 1(c). The corrected stop point labels yield an inferred occupied trip PIIID.

Uncovering trip occupancies from vehicle trajectories is also useful for a wide spectrum of downstream applications. For example, with a more complete knowledge of trips made by self-employed drivers based on both observed and inferred trips, ride-hailing services can gauge the demand for services and adjust the fare scheme accordingly. Another application is to use the observed and inferred trips to detect traffic issues at some locations. For example, a cluster of intermediate stop points may indicate traffic congestions (e.g., at specific road junctions) or long queue for dropping off or picking up passengers (e.g., at airports, CBD transit stations). To gain more insights from stop points, we may also perform spatial and temporal analyses to uncover spatial-temporal dynamics of each type of stop points.

Contributions. This paper makes the following contributions:



(a) Trip Occupancy

(b) Stop Point Classification

(c) Structural Trip Inference

Figure 1: Illustration of inferring unobserved trips

The LINT Framework. To address partial ground truth in vehicle trajectory, we propose a novel framework, namely, **L**earning to **IN**fer **T**rips (LINT), for inferring trips' occupancies based on the ideas of *stop point classification* and *structural trip inference*.

Stop Point Classification and Analysis. We develop a stop point classifier for learning stop points from occupied trips using our proposed point features. We reveal the economic activities and urban dynamics by analyzing geographical locations and temporal distributions of all the labeled stop points.

Cost-Effective Segmentation Algorithms. We devise several segmentation algorithms, including DP, GS and EGS, for generating valid label sequences from a sequence of predicted labeled stop points.

Experiment Evaluation using Vehicle Trajectory Data. Our experiments show that the proposed stop point classifier can accurately label stop points. We also demonstrate that GS achieves high accuracy with efficient running time.

In the rest of this paper, we present the LINT framework in Section 2. Sections 3 introduces our method for stop point classification. Section 4 describes proposed segmentation algorithms. Section 5 evaluates our LINT framework using real-world datasets. Section 6 reviews the related work. Finally, Section 7 concludes this study.

2 THE LINT FRAMEWORK

In this section, we give an overview of the LINT framework. LINT is designed to infer occupied and unoccupied trips from unobserved trips by exploiting a given set of observed occupied trips within a vehicle trajectory. Figure 2 depicts the LINT framework, which consists of two major components: (1) *stop point classification*, and (2) *structural trip inference*, which are detailed as follows.

Stop Point Classification. The first step of LINT learns stop point classifiers from the observed occupied trips by three sub-steps: stop point extraction, feature extraction, and classifier training. First, we extract stop points and their labels from the observed occupied trips. A stop point refers to a consecutive sequence of trajectory points indicating a near-stationary vehicle (i.e., speed is close to zero). Second, we explore and extract features associated with the stop points. Third, we train a classifier to determine pick-up, drop-off, and intermediate stop points using the proposed features.

Structural Trip Inference. The second step of LINT consists of three sub-steps: stop point/feature extraction, classification, and

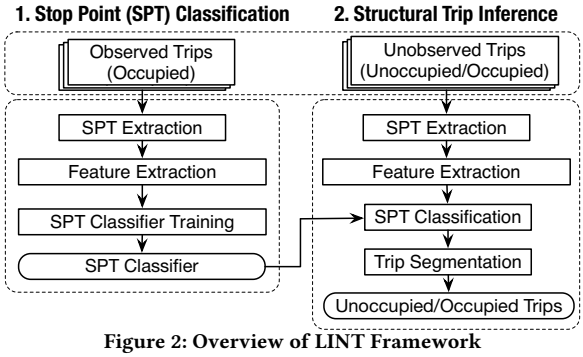


Figure 2: Overview of LINT Framework

Table 1: Statistics of our Ride-Hailing Dataset

Trajectory	Occupied	Unobserved
# trips	54,567	54,979
# sampling points($\times 10^6$)	5.3	35.9
travel distance($\times 10^6$km)	0.4	3.1
Stop Point	**Training Size**	**Testing Size**
intermediate	336,566 (0.825)	84,094 (0.825)
pick-up	38,002 (0.093)	9496 (0.093)
drop-off	33,538 (0.082)	8377 (0.082)

segmentation. Given a vehicle trajectory with unobsesrved trips, we first extract stop points and generate features for the extracted stop points. Second, we predict labels of extracted stop points using the stop point classifier trained from the stop point classification component. As a result, the vehicle trajectory of unobserved trips is transformed into a sequence of labelled stop points (or a sequence of labels, in short). Unfortunately, the sequence of labels may not follow the *structure validity constraint*, a pattern of occupied trips that begins with a pick-up stop point, followed by a number of intermediate stop points, before ending with a drop-off stop point. To address this issue, we propose segmentation algorithms which take a sequence of labelled stop points as input and make appropriate corrections to meet the structure validity constraint.

The major research issues in LINT framework lie in two aspects: (i) to effectively classify stop points, and (ii) to devise cost-effective segmentation algorithms under imposed constraints to infer trips. Both tasks are nontrivial. We discuss the challenges and propose solutions in following sections.

3 STOP POINT CLASSIFICATION

In LINT, an essential task for inferring occupied and unoccupied trips is to assign labels to stop points of a trajectory, which is formulated as a stop point classification problem.

Problem 3.1. (Stop Point (SPT) Classification) Given a set of occupied trips $C = \langle c_1, \cdots, c_m \rangle$, a speed threshold ϵ_s, and a stationary time threshold ϵ_d, learn a classifier to label a given stop point of unknown type as pick-up, intermediate, or drop-off.

3.1 Stop Point Extraction

Stop point is an aggregation of near by trajectory sampling points where the vehicle mostly stops and is formally defined as below.

Definition 1. (Stop Point) A stop point corresponds to a sequence of trajectory sampling points s=$\langle q_{i+1}, \cdots, q_{i+k} \rangle$ such that the speed between any consecutive points is less than the *speed threshold* ϵ_s and the time difference between the first and last point is less than a *stationary time threshold* ϵ_d. The location of the stop point is represented by the centroid of s, where $s.lat = \frac{1}{k}\sum_{j=1}^{j=k} q_{i+j}.lat$ and $s.lng = \frac{1}{k}\sum_{j=1}^{j=k} q_{i+j}.lng$. The time of the stop point is also derived by the mean time point in s, i.e., $s.t = \frac{1}{k}\sum_{j=1}^{j=k} q_{i+j}.t$.

The label of stop points from the observed occupied trips is determined as follows. For each occupied trip c_j corresponding to a booking b_j, we label the stop points covering the pick-up and drop-off points of c_j as *pick-up* and *drop-off* stop points, respectively. The remaining stop points between pick-ups and drop-offs are labeled as *intermediate* stop points. Accordingly, we obtain a set of stop points and their labels from the trajectories as depicted in Table 1. We observe that the majority of stop points (> 80%) are intermediate.

3.2 Feature Extraction

Given a target stop point s_i, We investigate three types of features: *spatial density*, *temporal density*, and *spatial-temporal density* of stop points. These features aim to capture *co-occurrences* of the same type among nearby stop points of s_i.

Spatial Density. Empirically, we observed that the spatial locality of stop points, i.e., stop points of the same type tend to share similar locations. For example, city landmarks are natural for pick-up and drop-off, while busy road junctions are likely intermediate stop points. In other words, the spatial correlation between s_i and l-type stop points (where $l \in$ { drop-off, pick-up, and intermediate}) is salient, if l-type stop points appear more often in the neighborhood of s_i than other types. To capture the pattern, we formally define the *l-type spatial density of stop point s_i* to be the popularity-neutralized ratio of l-type stop points to the number of stop points in the neighborhood of s_i as follows:

$$f_s(s_i, l) = \frac{|N(r, s_i) \cap S_l|}{|N(r, s_i)| \cdot |S_l|} \tag{1}$$

where S_l is the universal set of l-type stop points and $N(r, s_i)$ is the set of nearby stop points located within r radius of s_i. The $N(r, s_i)$ in denominator aims to derive the proportion of l-type stop points in the neighborhood of s_i, i.e., $|N(r, s_i) \cap S_l|/|N(r, s_i)|$. However, due to the dominance (> 80%) of intermediate stop points over all places, we further normalize the ratio by $|S_l|$ to neutralize the popularities of stop points. $f_s(s_i, l)$ returns a value between 0 and 1. A higher value of $f_s(s_i, l)$ indicates l-type is more dominant in the neighborhood of s_i.

Temporal Density. We also hypothesize that the temporal correlation between label types and stop points s_i is salient. For example, pick-ups/drop-offs are likely to appear more often during morning and evening peak hours. If l-type stop points appear more often within a time window ω from the $s_i.t$ than other types of stop points, it suggests a higher temporal correlation between l-type stop points and s_i. Thus, we define the *l-type temporal density of stop point s_i* to be the ratio of l-type stop points to the number of stop points within a time window ω from the $s_i.t$ as follows:

$$f_t(s_i, l) = \frac{|N(\omega, s_i) \cap S_l|}{|N(\omega, s_i)| \cdot |S_l|} \tag{2}$$

where $N(\omega, s_i)$ is the set of stop points within ω_i from s_i. Similarly, the $|S_l|$ in denominator neutralizes the popularities of stop points.

Spatial-Temporal Density. Finally, we measure the spatial-temporal correlation between l-type stop points and s_i. Likewise, we define the *l-type spatial-temporal density of stop point s_i* as follows:

$$f_{st}(s_i, l) = \frac{|N(\omega, r, s_i) \cap S_l|}{|N(\omega, r, s_i)| \cdot |S_l|} \quad (3)$$

where $N(\omega, r, s_i)$ is the set of stop points within ω (e.g., 30 mins) and r (e.g., 30 meters) away from s_i. The $|S_l|$ in denominator again neutralizes the popularities of stop points.

Our features are evaluated in a statistically sound manner. For example, we find the mean values of spatial-temporal density for drop-offs (0.987) against non-drop-offs (0.005) are significantly different, with p-value $< 1.0 \times 10^{-4}$ at the 0.05 level.[1] This coincides with typical clustering effect on types of stop points, where drop-offs are usually spatial-temporally co-located with drop-offs rather than with non-drop-off points.

3.3 Training Classifier

Once the stop points are extracted from the observed occupied trips, the point features are extracted to train stop point classifiers. We adopt one-vs-all strategy to classify each type of stop points. This gives rise to three one-vs-all classifiers: NI/I (i.e. non-intermediate vs intermediate stop points), P/NP (i.e. pick-up vs non-pick-up stop points), and D/ND (i.e. drop-off vs non-drop-off stop points). A stop point s_i is given to the three classifiers simultaneously to determine its label. Once we obtain the prediction outcomes from supervised learning models (e.g., using SVMs), we then transform the prediction outcomes to posterior probabilities using Platt Scaling [10, 11].

4 STRUCTURAL TRIP INFERENCE

Since the label prediction of stop points in a trajectory sequence is independent from one another, the predicted labels may violate the *structural validity constraint*, i.e., any occupied trip begins with a pick-up, followed by any number of intermediate stop points, before ending with a drop-off. Thus, we design the structural trip inference component in LINT for the following refinement task.

Problem 4.1. (Structural Trip Segmentation (STS)) Given an initial label sequence $Z_0 = \langle z_1, \cdots, z_T \rangle$, where z_i is the label of i^{th} stop point in the sequence ($z_i \in L = \{D, P, I\}$), find a refined label sequence $Z' = \langle z'_1, \cdots, z'_T \rangle$ of "*good quality*" such that Z' simultaneously satisfies the following structural validity constraints.

$$0 \le (num(P, Z[1, i]) - num(D, Z[1, i])) \le 1, \forall 1 \le i \le T \quad (4a)$$

$$0 \le (num(D, Z[i, T]) - num(P, Z[i, T])) \le 1, \forall 1 \le i \le T \quad (4b)$$

where $num(P, Z[1, i])$ returns the number of P labels in sequence $Z[1, i]$. Specifically, Eq. (4a) captures the *prefix validity constraint* that mandates the prefix sequence from the beginning up to i to have either equal number of P and D labels or one more P label than D. Eq. (4b) captures the *suffix validity constraints*. The suffix sequence from position i onwards must have either equal number of P and D labels or one more D label than P labels.

Problem Analysis. A naive approach to STS is to generate all possible label sequences, eliminate those not satisfying the validity constraints, and then return the one(s) with "good quality". This requires searching the entire solution space $O(3^T)$, which is not efficient. Additionally, there is no clear measure of the "good quality" mentioned above. Intuitively, the quality of a valid label sequence is good if it is closely similar to the true occupied trip. Unfortunately, we do not have ground truth to measure errors in the inference process. Thus, we propose several heuristics using the *probability profile* of a label sequence as a guidance to achieve our goal.

[1] Due to space constraint, we skip the discussion of spatial and temporal densities.

Definition 2. (Probability Profile) Given an label sequence Z, its probability profile matrix $M_{T\times|L|}$ consists of entries $M(i, z_i)$ denoting the probability that the i^{th} stop point being labeled as z_i.

The matrix entries $M(i, z_i)$ can be estimated based on posterior probabilities $p(z_i = x|s_i)$ (i.e.,) for the i^{th} point of Z to have label $x \in L$, using our SPT classifier. As the key challenge in the STS problem is to find a refined label sequence of (unmeasurable) good quality, the knowledge of probability profile $M_{T\times|L|}$ may potentially provide clues to find a label sequence that has few errors from the original stop point labels in the unobserved trajectory.

In the following, we formulate the STS task as an optimization problem based on the probability profile matrix, and then leverage dynamic programming to find an optimal label sequence.

4.1 Dynamic Programming (DP)

We measure the quality of a label sequence Z' by using *likelihood* defined in Eq. (5a) as a proxy. The goal of DP is to find a label sequence Z that (i) minimizes the log-likelihood of Z (see Eq. (5b)), and (ii) satisfies the structure validity constraint defined in Eq. (4).

$$p(Z') = p(z'_1, z'_2, \cdots, z'_T; M) = \prod_{i=1}^{T} p(z_i|M) = \prod_{i=1}^{T} M(i, z_i) \quad (5a)$$

$$\arg\min_{Z} \log(p(Z)) \quad (5b)$$

First, we notice that the aforementioned optimization problem is not eligible to employ dynamic programming given label types (P,I,D) because the property of optimal substructure does not hold. To address this issue, we introduce two types of intermediate points: (1) I_O (occupied): the intermediate points after P and before D, and (2) I_U (unobserved): the intermediate points after D and before P. We provide the proof of eligibility to use dynamic programming for the optimization problem given label types (P,I_O,D,I_U) as follows:

Proof. Let $Z[1,T]$ be the optimal label sequence from position 1 to T. Assume that this optimal label sequence contains I_O at i^{th} position, the solution thus can be split into $Z[1,i]$ and $Z[i,T]$. If there is a sequence Z' with better log-likelihood from 1 to i, $\log(p(Z'[1, i])) < \log(p(Z[1, i]))$, then $\log(p(Z'[1, i]))+\log(p(Z[i, T])) < \log(p(Z[1, i]))+\log(p(Z[i, T]))$. This contradicts to the first statement that $Z[1,T]$ is the optimal label sequence from position 1 to T. □

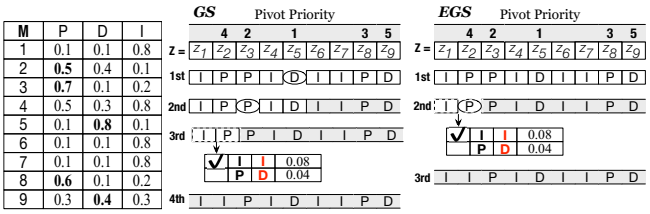
Once the property of optimal substructure holds, the optimal valid label sequence can be constructed from optimal solutions to its subsequences. Let $t(i,x)$ be the log-likelihood of an optimal subsequence up to i, where $x \in \{P, I_O, D, I_U\}$. Let $trans(x,y)$ be true if and only if it is valid to go from label x to label y. For example, the valid transitions to drop-off points include $trans(P,D)$=true and $trans(I_O,D)$=true. The invalid transition to drop-off points is $trans(I_U,D)$=false. The dynamic programming function incorporating validity constraints can be defined as follows:

$$t(i + 1, y) = \min_{trans(x,y)=\text{true}} t(i, x) + \log M(i + 1, y) \quad (6)$$

Our goal is then to find the minimum of t(T,D) or t(T,I_U) with the first position being either $t(1,P)$ or $t(1,I_U)$ such that validity constraints in Eq. (4) hold.

Table 2: Optimal Sequence $\langle P, D, P, I_O, D, I_U, I_U, P, D\rangle$

i	1	2	3	4	5	6	7	8	9
P	**2.3**	0.91	**1.51**	3.45	5.02	5.64	5.86	**4.29**	6.19
I_O	0.22	2.53	2.52	**3.12**	5.42	5.25	5.47	6.67	5.5
D	2.3	**1.15**	3.21	2.71	**3.34**	7.33	7.55	7.77	**5.21**
I_U	0.22	2.53	2.76	4.37	5.01	**3.56**	**3.78**	4.99	6.19



(a) Probability Profile (b) GS and EGS

Figure 3: Comparison of proposed algorithms.

Algorithm 1: TripSegmentation (GS and EGS)

Input:
$Z = \{z_1, \ldots, z_T\}$: initial label sequence, $M_{T\times|L|}$: probability profile of Z, α and λ: error and z-score threshold;

Output:
Z': the valid label sequence with maximum likelihood

1 $\lambda_Z \leftarrow$ z-score($|Z|$), $n_{NI} \leftarrow$ numNI(Z);
/* base case: make corrections */
2 **if** $\lambda_Z \leq \lambda$ & $n_{NI} \leq \alpha$ & $\neg valid(Z)$ **then return** $CorrectSeq(Z,M,\alpha)$;
/* recursion case: division */
3 **if** EGS **then**
4 $\quad Z_{lv}, Z_{rv} \leftarrow$ findValidPrefixSuffix(Z);
5 $\quad Z \leftarrow Z - Z_{lv} - Z_{rv}$;
6 **else if** GS **then**
7 $\quad Z_{lv} \leftarrow \varnothing, Z_{rv} \leftarrow \varnothing$;
8 $pivot \leftarrow$findPivot(Z);
9 $Z_l \leftarrow Z[low, pivot]$; $Z_r \leftarrow Z[pivot+1, high]$;
10 **if** $\neg valid(Z_l)$ **then** Z'_l = TripSegmentation(Z_l,M,α,λ) **else** $Z'_l = Z_l$;
11 **if** $\neg valid(Z_r)$ **then** Z'_r = TripSegmentation(Z_r,M,α,λ) **else** $Z'_r = Z_r$;
12 **return** $Z_{lv} \oplus Z'_l \oplus Z'_r \oplus Z_{rv}$;

Running Example. Consider the sequence in Figure 3, DP builds a table for each possible label subsequences as illustrated in Table 2. Each entry $t(i,x)$ indicates the log-likelihood of obtaining a subsequence up to i, Z[1,i], ending with label $x \in \{P, I_O, D, I_U\}$. Starting from the last position, $t(i$=9,D) has the minimum log-likelihood. We then assign "D" to point at i=9. The valid transition from i=8 to $t(i$=9,D) includes $t(i$=8,P)=4.29 and $t(i$=8,I_O)=6.67. We then assign "P" to z_8 because it has the minimum log-likelihood among valid transitions. Moving on in the same manner to the very beginning of the sequence, the valid transitions given $t(i$=2,D) are $t(i$=1,P)=2.3 and $t(i$=1,I_O)=0.22. We assign "P" over "I_O" to z_1 even if $t(i$=1,I_O) has smaller log-likelihood in order to meet the constraint in Eq, (4). The final corrected sequence with two corrections is $\langle \mathbf{P}, \mathbf{D}, P, I_O, D, I_U, I_U, P, D\rangle$.

Complexity Analysis of DP. The time complexity to compute all $t(T$,x) is $O(4T)$. DP essentially builds a look-up table of size 4-by-T for a sequence of length T and four different point types.

4.2 Greedy Segmentation (GS)

The key idea of GS is to *secure labels of high likelihoods while prioritizing corrections based on their likelihoods* rather than trivially optimizing global likelihood (as DP does). GS adopts a divide-and-conquer approach. When Z is structurally invalid but of "manageable" length and "simple" to correct, GS directly performs a correction on it to return a valid sequence Z'. Otherwise, GS divides Z into two subsequences Z_l and Z_r based on a carefully chosen *pivot* label. It then recursively correct Z_l and Z_r and returns the concatenation of Z'_l and Z'_r.

As shown in Algorithm 1, GS first determines if the given sequence Z is of manageable length and simple (Line 2). We say that a sequence is manageable if it includes a number of stop points almost identical to that of a typical occupied trip. We thus compute the z-score of the length of sequence Z and consider Z to be manageable if the z-score is less than a threshold λ. Let the mean and standard deviation of the number of stop points in observed occupied trips be denoted by μ and σ, respectively. The z-score of $|Z|$, λ_Z, is computed by $\frac{|Z|-\mu}{\sigma}$. The μ and σ of observed occupied trips are 9.9 and 3.7, respectively, in our dataset. On the other hand, Z is considered simple if it does not have too many P and D labels, which are the labels to be corrected. We use an error threshold α to determine a simple sequence. A manageable and simple sequence is corrected by invoking the *CorrectSeq* procedure.

Choice of Pivot. A non-manageable or non-simple sequence has to be further divided into Z_l and Z_r subsequences using a pivot (Line 8-9). We select the position idx in the sequence with most confident P or D label based on the profile matrix, as the pivot represents the beginning or end of some occupied trips which are least likely to be corrected (i.e., $idx = \arg\max_{i, z_i \in \{\text{P,D}\}} M(i, z_i)$). If Z_l (or Z_r) is structurally invalid, we recursively call Algorithm 1 with the appropriate parameters including error and z-score threshold to correct it (Line 10-11). Finally, we return the concatenation of Z'_l and Z'_r (Z_{lv} and Z_{rv} are $\varnothing$ in Line 12).

Selection of Candidate Corrections. The *CorrectSeq* procedure aims to find a correction to a given label sequence. Ideally, we prefer only a few label changes to minimize loss in the likelihood of label sequence. We thus use a priority queue Q to efficiently maintain candidate label changes ordered by decreasing likelihood value.

Starting from the candidate of label change with the largest likelihood value, we perform one single change on Z to obtain Z_c. If Z_c is valid, we add it to our valid candidate set, otherwise it is added to invalid candidate set. For the invalid candidates, more changes may be further made to generate valid candidates as long as the number of changes is capped by α. Finally, we return the valid candidates with the maximum likelihood.

Generating valid candidates with α changes requires $\sum_{i=1}^{\alpha} C_i^{|Q|}$ examinations. We generate candidates incrementally until α label changes are explored. Note that α is bounded by number of P's and D's in the sequence. The likelihood of a sequence does not strictly decrease as more corrections are made. Thus, we have no choice but to explore candidates with more corrections (up to α) even when a feasible solution with fewer corrections is found.

Likelihood Pruning Strategy. If a sequence has maximal likelihood, each of its non-overlapping subsequences also has maximum likelihood. We thus maintain a heap to sort valid candidates Z_c's by their local likelihood p(Z_c)'s and return the one with maximum local likelihood as the corrected sequence.

Running Example. Figure 3(b) illustrates GS. Consider the sequence in Figure 3(b) with α=2 and λ=2. The sequence is not simple as it contains more than α NI labels. Hence, GS divides it into two subsequences $Z[1,5]$ and $Z[6,9]$. GS selects z_5=D as the pivot due to its highest probability 0.8 among all the NI points. $Z[6,9]$ is valid (shaded) and thus requires no more work. On the other hand, $Z[1,5]$ is invalid and still not simple. It is further divided into $Z[1,2]$, and $Z[3,5]$ using z_3=P as the pivot. $Z[3,5]$ is valid and remains intact. Since $Z[1,2]$ contains no more than α=2 NI labels, direct error correction is performed on it. Between two valid candidates "II" and "PD", "II" (likelihood("II")=0.08) is chosen due to a larger likelihood. The final corrected sequence with only one correction is $\langle I, \mathbf{I}, P, I, D, I, I, P, D\rangle$.

Complexity Analysis of GS. The validity examinations require $O(T \log T)$ time for a sequence of length T. GS at each base case examines $\sum_{i=1}^{\alpha} C_i^{k\times|L|}$ candidates with length k, $\alpha = 2 \ll T$ and returns the valid candidate with the maximal likelihood. Thus, each base case requires $O(k^2)$ time for candidate generation. In summary, the time complexity of GS is $O(T \log T \times k^2)$.

4.3 Efficient Greedy Segmentation (EGS)

To further enhance the efficiency of GS, we propose **E**fficient **G**reedy **S**egmentation (EGS) to efficiently derive feasible solutions without compromising too much optimality. The key idea here is to keep valid subsequences untouched (if possible) before diving into refinement. As such, the refinement area can be greatly minimized.

Valid Prefix and Suffix Pruning. Given sequence $Z[low,high]$, let Z_{lv}=$Z[low,lh]$ (also called the *valid prefix* of Z) be the left most maximal valid subsequence of Z, where $lh \leq high$. Similarly, let $Z_{rv} = Z[rh, high]$ (called the *valid suffix* of Z) be the right most maximal valid subsequence of Z. The remaining subsequence, $Z[lh,rl]$, where $lh \leq rl$, is called the *Minimal Invalid Sequence*.

As candidate generation for base cases (Algorithm 1, Line 2) is the most computationally intensive step in GS, we introduce a strategy to minimize the examination of subsequences. To achieve this, EGS first identifies the most "minimal invalid subsequence" by eliminating valid prefix and suffix in a given sequence in each recursion step. The pseudo code of EGS is given in Algorithm 1. In the recursion case, EGS first finds the valid prefix and the valid suffix (Line 4). Afterwards, EGS processes the remaining invalid sequence Z (Line 5) like what GS does in its recursion phase. EGS finds a pivot from the updated Z and then divides the latter into Z_l and Z_r (Line 8-9). Depending on the validity of Z_l and Z_r, more recursion may be required (Line 10-11). In the base case, EGS breaks the recursion if the sequence satisfies the manageability and simplicity conditions, or searches for the valid candidate with maximal likelihood otherwise.

Running Example. Figure 3(b) illustrates a running example by EGS to showcase the efficiency of EGS. Given a sequence Z=IPPIDIIPD, EGS detects the valid suffix $Z[3,9]$ (shaded). There is no valid prefix. EGS thus focuses on correcting $Z[1,2]$. Similar to GS, the likelihood pruning strategy prunes away candidate "II", selecting the best candidate sequence "PD" for $Z[1,2]$. The final corrected sequence with only one correction is $\langle I, \mathbf{I}, P, I, D, I, I, P, D\rangle$.

Complexity Analysis of EGS. EGS performs in $O(T)$ attempts to examine the validity of a sequence of length T. For each base case, EGS examines $\sum_{i=1}^{\alpha} C_i^{3k}$ candidates of length k, $\alpha = 2 \ll T$ and reports the valid candidate with the maximal likelihood. Thus, each base case takes $O(k^2)$ time for candidate generation. In summary, the time complexity is $O(T \times k^2)$.

5 PERFORMANCE EVALUATION

In this section, we evaluate our LINT framework using real datasets and report the experimental results and findings.

5.1 Datasets and Settings

5.1.1 Datasets. We acquired a dataset from a ride-hailing mobile app provider.[2] The first part of the dataset, **booked trips data**, consists of millions of booking records, which contain pick-up and drop-off locations as well as the time of bookings made by users in Singapore from July to September 2014 via the app. The second part of the dataset, **vehicle trajectory data**, covers the GPS trajectories of 100 active self-employed drivers on the app during the same period, with a sampling interval of about 15 seconds. We also obtained the city's **road network** from OpenStreetMap (OSM).

Given vehicle trajectories and bookings, we first clean the data by removing duplicate bookings made by the same users within a short period, and noisy trajectory points using Map-matching techniques [2]. Second, we determine the pick-up and drop-off stop points of *observed occupied trips* using both trajectory and booking data. Finally, we extract all stop points with given speed threshold ϵ_s=5 km/h and stationary time threshold ϵ_d=10 min.

From the trajectory and booking data, we obtain 336,566 (82.5%) intermediate, 38,002 (9.3%) pick-up and 33,538 (8.2%) drop-off stop points from 54,567 occupied trips which serve as the ground truth for training and testing of the stop point classifiers.[3] In experiments, we use the trajectories covering 80% observed occupied trips for training and the remaining trajectories, i.e., the other 20% of observed occupied trips, for testing. Through a training and testing process, we harvest the stop point classifiers and the LINT framework for deployment. Finally, all unobserved trajectories, which consist of both occupied trips and unoccupied trips, are further processed and analyzed using the deployed LINT framework.

To train stop point label classifiers, we use SVM [5] and logistic regression (LR) models, which are evaluated via 5-fold cross validation. We then apply DP, GS, and EGS on stop point label sequences to infer valid sequences for occupied and unoccupied trips.

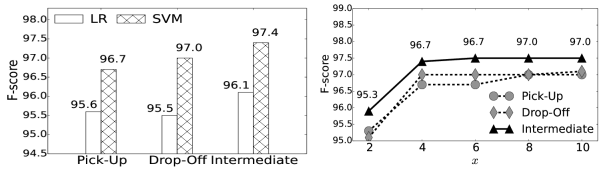
5.1.2 Evaluation Metrics.

Stop Point Classification. In this experiment, we aim to answer three research questions: (1) how effective and robust are the proposed feature types? (2) what is the impact of *imbalance ratio* between positive and negative samples?, and (3) what are the spatial-temporal properties of each type of stop points? We measure the accuracy of stop point classification using the following metrics.

F-Score: $F\text{-}score(P) = 2 \cdot Pre(P) \cdot Rec(P)/(Pre(P) + Rec(P))$ where P is one of the label types (e.g., pick-up); $Pre(P)$ and $Rec(P)$ denote the precision and recall of prediction on class P, respectively, by a classifier under examination.

[2]We could not reveal the app name and detailed statistics due to non-disclosure agreement.

[3]The percentages of pick-up and drop-off stop points are not the same due to the missing drop-off labels in some bookings in our dataset.



(a) Classification Accuracy (b) Imbalance Ratio(x)

Figure 4: Performance comparison.

Structural Trip Inference. In this experiment, we aim to answer two research questions: (1) how well can our segmentation algorithms uncover the occupied trips? and (2) how many changes does each algorithm made upon the original sequence? We use *edit distance* and *correction rate* for the two questions respectively.

Edit distance (ED): Consider a ground truth sequence Z^{GT} under a mixed occupied and unobserved trips (e.g., GT in Table 4), and a predicted label sequences Z'_i. The edit distance is defined by the sum of unoccupied stop points in Z'_i that are occupied in Z^{GT}, and unobserved stop points in Z^{GT} that are occupied in Z'_i. Edit distance gives a total cost to transform Z'_i to Z^{GT}.

Correction rate (CR): This measures the fraction of corrections made to the original label sequence (i.e. #corrected labels in Z') over the size of the original label sequence Z. Note that correction rate is stricter than editing distance. A sequence may have zero editing distance but suffered from high correction rate when one occupied trip is mistakingly predicted as multiple occupied trips.

5.2 Evaluation on Stop Point Classification

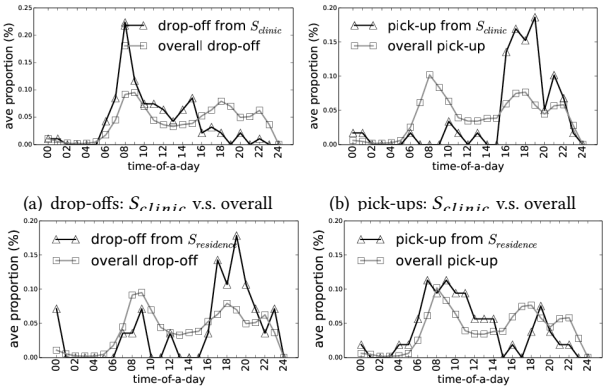
Effectiveness of Features. Figure 4(a) shows the effectiveness of the proposed point features across classification models (LR v.s. SVM). SVM outperforms LR for all 3 stop point labels and it achieves 96.7, 97.0, and 97.4 in F-score over pick-up, drop-off and intermediate classes. We also find that the most discriminative feature in NI/I classification is the spatial-temporal density of pick-ups. This matches our intuition that pick-up points are usually spatial-temporally closer to NI stop points rather than I stop points.

Impact of Imbalance Ratio. As there are many more I stop points than P/D stop points, the negative instances significantly out-number positive instances for P/NP, D/ND and NI/I classification, causing concerns on the issues of data imbalance and prolonged training time. We therefore study the impact of imbalance ratio x = number of negative instances / number of positive instances. Suppose pick-ups are the positive instances. We then use all of them n_p as positive instances, and $x \cdot n_p$ other stop points, i.e., intermediate and drop-off, to train the P/NP classifier. We study the impact of imbalance ratio using SVM classifier for $x \in \{2, 4, 6, 8, 10\}$. Figure 4(b) suggests that increasing x for negative instances improves classification accuracy for all stop point labels. Figure 4(b), the I/NI classifier, sees improvement of F-score from 0.953 (x=2) to 0.97 (x=10). Note that at x=2, the classifiers still perform extremely well (0.953) even though we down-sample the negative instances significantly.

Spatial-Temporal Properties of Stop Points. Given the set of classified stop points obtained using our classifier, we investigate the spatial properties of stop points by analyzing popular landmark types in the neighborhood of each type of stop points. We thus propose *popularity-neutralized ratio of t-type landmarks* in the

Table 3: Comparison of Stop Points Distribution

Landmarks	Clinic			Residence			Transit Station		
(%)	D	P	I	D	P	I	D	P	I
Global	8.2	9.3	82.5	8.2	9.3	82.5	8.2	9.3	82.5
Local	**31.6**	19.6	48.8	9.6	**17.6**	72.8	8.3	4	**87.7**



(a) drop-offs: S_{clinic} v.s. overall (b) pick-ups: S_{clinic} v.s. overall

(c) drop-offs: $S_{residence}$ v.s. overall (d) pick-ups: $S_{residence}$ v.s. overall

Figure 5: Temporal Distributions of Stop Points.

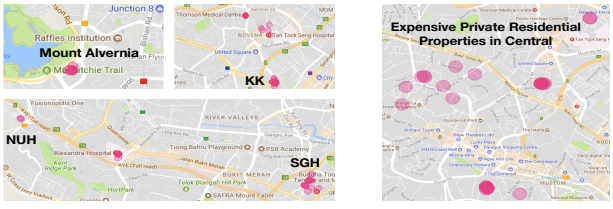
neighborhood of the stop point s_i as follows:

$$f_{lm}(s_i, t) = \frac{|N_{lm}(r, s_i) \cap L_t|}{|N_{lm}(r, s_i)| \cdot |L_t|} \tag{7}$$

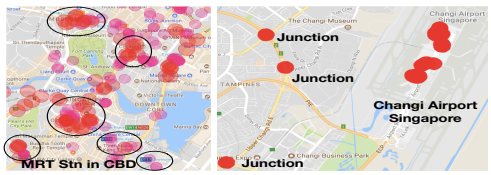
where $N_{lm}(r, s_i)$ denotes the set of landmarks within r (e.g., 50 meters) from the stop point s_i and L_t denotes the set of t-type landmarks in facebook business entries dataset. Due to the dominance of some landmarks, we further normalize the ratio by $|L_t|$ to neutralize the popularities of landmarks. $f_{lm}(s_i, t)$ returns a value between 0 and 1. A higher value of $f_{lm}(s_i, t)$ indicates t-type is more dominant in the neighborhood of s_i.

Popularity-neutralized ratio helps uncover stop points with exceptionally high or low concentration of certain type of landmarks. First, we select 300 stop points with the highest popularity-neutralized ratio for each type of landmark type t, denoted as S_t. We then derived the *local distribution* of stop point type from S_t to compare with the *global distribution*, i.e., the distribution of overall classified stop points: 84,094 (82.5%) intermediate, 9,496 (9.3%) pick-up, and 8,377 (8.2%) drop-off points. Lastly, we identify those landmark types t such that the alternative distribution drawn from S_t displays significant deviation from the global distribution. Through our analysis, three landmark types are identified as particularly interesting: $t \in$ {"clinic","residence","transit"}. Table 3 summarizes the deviations between the global distribution and three local distributions drawn from S_{clinic}, $S_{residence}$, and $S_{transit}$, respectively. For example, we observe a notable increase of drop-offs (31.6% v.s. 8.2%) in the neighborhoods of S_{clinic} against global distribution. This indicates a higher tendency of drop-offs in the neighborhoods of these clinics. We also observe notable increases in pick-ups (17.6% v.s. 9.3%) and intermediate stops (87.7% v.s. 79%) in the neighborhoods of $S_{residence}$ and $S_{transit}$, respectively.

We further investigate the temporal variations of stop points in S_{clinic}, and $S_{residences}$. Figure 5(a) shows the temporal distribution of (1) drop-off proportion from S_{clinic} and (2) overall drop-off proportion. We observe a much higher proportion of drop-offs against overall drop-off proportion during daytime [05:00,15:00]. Figure 5(b) shows the distribution of (1) pick-up proportion from



(a) drop-offs in S_{clinic} (b) pick-ups in $S_{residence}$



(c) intermediates in $S_{transit}$

Figure 6: Spatial Distributions of Stop Points (Best Viewed in Color).

S_{clinic} and (2) overall pick-up proportion. We observe a much higher proportion of pick-ups against overall pick-up proportion during night time [16:00,22:00]. Similarly, we observe a much higher proportion of drop-offs from $S_{residence}$ during night time [17:00,21:00] in Figure 5(c) and a much higher proportion of pick-ups from $S_{residence}$ during daytime [04:00,14:00] in Figure 5(d).

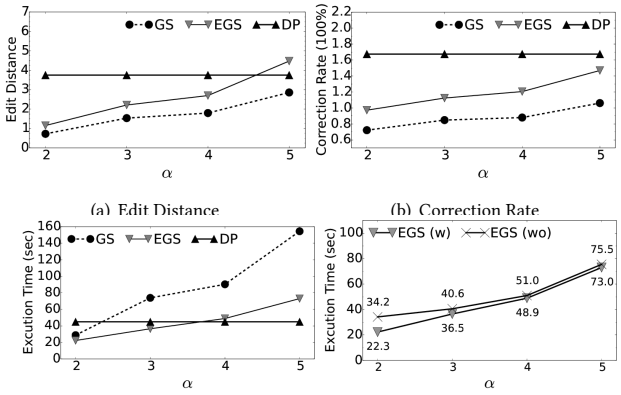
By coupling with landmark data, we perform spatial and temporal analyses and have interesting observations. For example, Figure 6(a) reveals that the drop-offs from S_{clinic} mostly cluster at major hospitals or medical centers in Singapore, such as Singapore general hospital (SGH), national university hospital (NUH), KK women's and children's hospital (KK). The pick-ups from $S_{residence}$ are mostly from central and east coast neighborhoods where expensive private residential properties are located as shown in Figure 6(b). Lastly, the intermediate points from $S_{transit}$ are mostly nearby road junctions or public transit places in Figure 6(c). The areas where enormous intermediate stop points cluster may indicate traffic congestions (e.g., at road junctions) or long queue for dropping off or picking up passengers (e.g., at airports, CBD transit stations).

5.3 Evaluation on Structural Trip Inference

We evaluate the performance of our segmentation algorithms in two aspects: (1) quality, and (2) efficiency.

Occupancy Recovery. Figure 7(a) shows how well the proposed structural segmentation algorithms can recover the ground truth in terms of edit distance as we vary threshold of correction numbers α. Figure 7(a) suggests that GS and EGS take one to four edit operations to recover ground truth per trajectory under various settings of α, while DP consistently requires approximately four edit operations from ground truth occupancy status. GS yields the least editing operations against ground truth. In particular, GS gives the most accurate trip segments when α=2 (edit distances = 0.7 on average). This suggests that prioritizing stop point labels is critical for accurate prediction of occupancy status.

Occupied Trip Recovery. Figure 7(b) shows that GS returns the most accurate stop point label sequence against ground truth sequence with varying error threshold α. When α=2, GS (0.72%) only requires half of the corrections (CR) required by DP (1.68%) while achieving smaller editing distance to recover occupancy status of



(a) Edit Distance (b) Correction Rate

(c) Execution Time (d) Execution Time

Figure 7: Performance comparison of segmentation results.

trajectories. This suggests that GS and EGS employ the knowledge of probability profile more effectively than DP.

Effectiveness of Pivot Selection Strategy. Figures 7(a) and 7(b) together suggest that prioritizing point labels may compromise global likelihood for a corrected label sequence that is more similar to the ground truth occupied trips. Pivot selection prioritizes valid substrings of high likelihood values, which enables GS and EGS to focus on invalid subsequences with small likelihood values. On the whole, the STS task greatly benefits from the pivot selection strategy and the divide-and-conquer framework.

Effect of α Threshold in Efficiency. Figure 7(c) compares the efficiency of GS, EGS, and DP. Among them, EGS incurs significantly less execution time compared to GS especially when α becomes larger. For example, the time taken by GS is 2.1 times the time taken by EGS at α=5. This is because the divide-and-conquer framework can efficiently locate invalid subsequences and safely avoid examining valid subsequences. DP achieves the best efficiency both theoretically $O(T)$ and empirically.

Effects of Likelihood Pruning Strategies. In Figure 7(d), EGS(wo) without likelihood pruning strategy incurs redundant computation to derive the final corrected label sequence. In contrast, EGS(w) benefits from the pruning strategy and incurs less execution time.

Case Studies on Inferring Trip Occupancies. Table 4 shows an example of how GS benefits from the pivot selection strategy compared to DP that optimizes likelihood. GT indicates the ground truth label sequence with two unobserved stop points and *Occupancy*(GT) is the occupancy status of GT. GS makes two corrections (CR=2) at $i = 8$ (I to D) and $i = 9$ (D to I), respectively; yielding *Occupancy*(GS) with only one edit distance to *Occupancy*(GT) at $i = 9$. DP also makes two corrections at $i = 3$ (P to D) and $i = 8$ positions (I to P); yielding *Occupancy*(DP) with seven edit distances to *Occupancy*(GT).

6 RELATED WORKS

Urban Computing. There has been significant amount of work on urban computing recently [22], including human mobility modelling [3, 6, 7, 17], urban planning in transportation [12, 13, 18–20, 24], traffic route prediction [1, 9, 16, 19, 20], etc. A number of applications driven by urban dataset, such as travel cost estimation [4, 14, 15] and refuel behavior sensing [21], have emerged.

Table 4: Comparison of trip occupancy in edit distance (ED) and correction rate (CR) (0, –, and ? strand for unoccupied, occupied, and unobserved).

i	1	2	3	4	5	6	7	8	9		
GT	?	?	P	I	I	I	I	I	D		
Occupancy(GT)	?	?	–	–	–	–	–	–	–		
Method	**Inferred Trip Occupancies**									**ED**	**CR**
GS	**I**	I	P	I	I	I	I	D	**I**		
Occupancy(GS)	0	0	–	–	–	–	–	–	0	1	2
DP	P	I_O	**D**	I_U	I_U	I_U	I_U	**P**	D		
Occupancy(DP)	–	–	–	0	0	0	0	0	–	7	2

Zheng et al. propose a travel time estimation model for any path based on trajectories of vehicles [14]. Liu et al. use taxi trajectory data to learn drivers' routing decisions [8]. Yuan et al. propose a recommendation framework for both taxi drivers and passengers based on passenger mobility patterns and taxi driver behavior from their GPS trajectories [20]. The paper also presents a probabilistic model to detect the occupied/cruising/parked status of a trajectory segment for a working taxi.

Trajectory Segmentation. One major issue often appeared in urban computing applications is trajectory segmentation [23][25]. Recently, Zheng et al. propose a supervised learning approach (decision tree, CRF etc.) to partition trajectory into segments of different transportation modes (e.g., bike/bus/driving/walk) [23]. With given transportation modes of trajectories, they propose to utilize supervised learning approaches to infer transportation modes based on segment features. Lastly, a postprocessing is introduced to improve the inferred sequences of transportation modes by incorporating typical user behaviors. Zhu et al. adopt a similar framework to infer the status of taxi trajectories [25]. The authors also resort to supervised learning approaches to infer the taxi status for each trajectory points given both ground truth of occupied and unoccupied taxi trajectories.

The ground truth labels, e.g., occupied and unoccupied labels of segments in [25], are given in all the aforementioned related works. This facilitates effective supervised learning for the targeted problems. Uniquely different from these previous works, we have only knowledge of a small subset of occupied trips. We therefore face a prediction task with *single class* ground truth due to lack of knowledge in unoccupied trips. The main innovation of this paper is thus to address posed challenges in prediction method and accuracy evaluation based on the ideas of *stop point classification* and *structural trip inference*.

7 CONCLUSIONS

We address a novel problem of inferring unobserved trips from vehicle trajectories and booking data using a small set of occupied trips. We propose a new framework, Learning to INfer Trips (LINT), for inferring occupied and unoccupied trips based on the ideas of stop point classification and structure trip inference. For the former, we extract point features to effectively classify stop points. In addition, we propose to use probability profiles as potential clues to find a valid label sequence of "*good quality*" for the latter. We develop a dynamic programming (DP) algorithm to infer trip occupancies from stop point label sequences. Additionally, we propose novel greedy segmentation (GS) and its efficient variant greedy algorithm (EGS). We conduct extensive experiments to evaluate our proposals. Results show that stop point labels can be predicted with high F-score. GS achieves high accuracy with efficient running time.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative. Wang-Chien Lee's work is supported in part by National Science Foundation under Grant No. IIS-1717084.

REFERENCES

[1] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z He, and Y. Li. 2017. Finding lowest-cost paths in settings with safe and preferred zones. *VLDB Journal* 26, 3 (2017).
[2] Guoyong Cai, Rui Lv, Liyuan Wang, and Hao Wu. 2014. A Novel Map Matching Algorithm. *Applied Mechanics & Materials* 556 (2014), 4139–4145.
[3] E. Cho, S. A. Myers, and J. Leskovec. 2011. Friendship and Mobility: User Movement in Location-based Social Networks. In *SIGKDD*.
[4] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu. 2016. Path Cost Distribution Estimation Using Trajectory Data. *Proceedings of the VLDB Endowment* 10, 3 (2016).
[5] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research* 9 (2008), 1871–1874.
[6] F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. 2011. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *VLDB Journal* 20, 5 (2011).
[7] M. Lichman and P. Smyth. 2014. Modeling Human Location Data with Mixtures of Kernel Densities. In *SIGKDD*.
[8] Siyuan Liu, Shuhui Wang, Ce Liu, and Ramayya Krishnan. 2015. Understanding taxi drivers' routing choices from spatial and social traces. *Frontiers of Computer Science* 9, 2 (2015).
[9] Y. Liu, C. Liu, N. J. Yuan, L. Duan, Y. Fu, and H. Xiong. 2014. Exploiting Heterogeneous Human Mobility Patterns for Intelligent Bus Routing. In *ICDM*.
[10] Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting Good Probabilities with Supervised Learning. In *ICML*.
[11] John C. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*.
[12] M. Qu, H.-S. Zhu, J. Liu, G. Liu, and H. Xiong. 2014. A Cost-effective Recommender System for Taxi Drivers. In *SIGKDD*.
[13] Masamichi Shimosaka, Keisuke Maeda, Takeshi Tsukiji, and Kota Tsubouchi. 2015. Forecasting Urban Dynamics with Mobility Logs by Bilinear Poisson Regression. In *UbiComp*.
[14] Y. Wang, Y. Zheng, and Y. Xue. 2014. Travel Time Estimation of a Path using Sparse Trajectories. In *SIGKDD*.
[15] B. Yang, C. Guo, and C. S. Jensen. 2013. Travel Cost Inference from Sparse, Spatio Temporally Correlated Time Series Using Markov Models. *Proceedings of the VLDB Endowment* 6, 9 (2013).
[16] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang. 2014. Multi-cost optimal route planning under time-varying uncertainty. In *ICDE*.
[17] N. Yang, X. Kong, F. Wang, and P. S. Yu. 2014. When and Where: Predicting Human Movements Based on Social Spatial-Temporal Events. In *SDM*.
[18] J. Yuan, Y. Zheng, and X. Xie. 2012. Discovering regions of different functions in a city using human mobility and POIs. In *SIGKDD*.
[19] J. Yuan, Y. Zheng, C. Zhang, W.-L. Xie, X. Xie, G. Sun, and Y. Huang. 2010. T-drive: Driving Directions Based on Taxi Trajectories. In *SIGSPATIAL*.
[20] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. 2013. T-finder: A recommender system for finding passengers and vacant taxis. *TKDE* 25, 10 (2013).
[21] F. Zhang, N. J. Yuan, D. Wilkie, Y. Zheng, and X. Xie. 2015. Sensing the Pulse of Urban Refueling Behavior: A Perspective from Taxi Mobility. *ACM TIST* 6, 3 (2015).
[22] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM TIST* 5, 3 (2014).
[23] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. 2010. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web (TWEB)* 4, 1 (2010), 1.
[24] C. Zhong, S. M. Arisona, X. Huang, and G. Schmitt. 2013. Identifying spatial structure of urban functional centers using travel survey data: a case study of Singapore. In *SIGSPATIAL International Workshop on Computational Models of Place*.
[25] Yin Zhu, Yu Zheng, Liuhang Zhang, Darshan Santani, Xing Xie, and Qiang Yang. 2012. Inferring taxi status using gps trajectories. *arXiv preprint arXiv:1205.4378* (2012).