Efficiently Estimating Statistics of Points of Interests on Maps

Pinghui Wang, Wenbo He, and Xue Liu

Abstract—Recently, map services (e.g., Google maps) and location-based online social networks (e.g., Foursquare) attract a lot of attention and businesses. With the increasing popularity of these location-based services, exploring and characterizing points of interests (PoIs) such as restaurants and hotels on maps provides valuable information for applications such as start-up marketing research. Due to the lack of a direct fully access to PoI databases, it is infeasible to exhaustively search and collect all PoIs within a large area using public APIs, which usually impose a limit on the maximum query rate. In this paper, we propose sampling methods to accurately estimate PoI statistics such as sum and average aggregates from as few queries as possible. Experimental results based on real datasets show that our methods are efficient, and require six times less queries than state-of-the-art methods to achieve the same accuracy.

Index Terms—Points of interests, sampling, measurement

1 Introduction

Aggregate statistics (e.g., sum, average, and distribution) of points of interests (PoIs), e.g., restaurants and hotels on map services such as Google maps [2] and Foursquare [3], provide valuable information for applications such as marketing decision making. For example, the knowledge of the PoI rating distribution enables us to evaluate a particular PoI's relative service quality ranking. Moreover, a restaurant start-up can infer food preferences of people in a geographic area by comparing the popularity of restaurant PoIs serving different cuisines within the area of interest [4]. Meanwhile, it can also estimate its market size based on PoI aggregate statistics, such as the number of Foursquare users checked in PoIs within the area. Similarly, a hotel start-up can utilize hotel PoIs' properties such as ratings and reviews to understand its market and competitors.

To exactly calculate the above aggregate statistics, it requires to retrieve all PoIs within the area of interest. However most map service providers do not provide the public with a direct fully access to their PoI databases, so we can only rely on public map APIs to explore and collect PoIs. Moreover, public APIs usually impose limits on the maximum query rate and the maximum number of PoIs returned in a response to a query, therefore it is costly to collect PoIs within a large area. For example, Foursquare map API [5] returns up to 50 PoIs per query and it allows 500 queries per hour per account. To collect PoIs within 14 cities in Foursquare, Li et al. [6] spent almost two months using 40 machines in parallel.

- *P. Wang is with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, Shaanxi, China. E-mail: phwang@mail.xjtu.edu.cn.*
- *W. He and X. Liu are with the School of Computer Science, McGill University, QC, Canada. E-mail: {wenbohe, xueliu}@cs.mcgill.ca.*

Manuscript received 8 Dec. 2013; revised 23 Oct. 2014; accepted 1 Sept. 2015. Date of publication 22 Sept. 2015; date of current version 6 Jan. 2016. Recommended for acceptance by X. Lin. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TKDE.2015.2480397

To address the above challenge, *sampling* is required. That is, a small fraction of PoIs are sampled and used to calculate PoI statistics. Due to the lack of a direct fully access to PoI databases, one cannot sample over PoIs in a direct manner, so it is hard to sample PoIs uniformly. The existing sampling methods [7], [8] have been proved to sample PoIs with biases. After sampling a fraction of PoIs using these two methods, one has no guarantees whether the PoI statistics obtained directly are to be trusted. To solve this problem, Dalvi et al. [7] propose a method to correct the sampling bias. However the method is costly because it requires a large number of queries for each sampled PoI (e.g., on average 55 queries are used in their paper). The method in [8] samples PoIs with unknown bias, so it is difficult to remove its sampling bias.

In this work we propose a new method random region zoom-in (RRZI) to eliminate the estimation bias. The basic idea behind RRZI is to sample a set of sub-regions from an area of interest at random and then collect PoIs within sampled regions. However, when we query a sampled sub-region including a large number of PoIs, an unknown sampling bias is introduced if we only collect PoIs returned. Otherwise, we need to further divide the sampled sub-region to exhaustively collect all PoIs within it. It requires a large number of queries. To solve this problem, we divide the area of interest into fully accessible sub-regions without overlapping, where a region is defined as a fully accessible region if it includes PoIs less than the maximum number of PoIs returned for a query. Then it is efficient to collect PoIs within a sampled sub-region, which requires just one query. To sample a fully accessible region, RRZI works as follows: From a specified area, RRZI divides the current queried region into two sub-regions without overlapping, and then randomly selects a non-empty sub-region as the next region to query. It repeats this process until it observes a fully accessible region. We show that RRZI is efficient, and it requires only a few queries to sample a fully accessible region. Besides its efficiency, the sampling bias of RRZI is easy to be corrected, which requires no extra queries in

TABLE 1
Overview of Baidu PoI Datasets

Area	Latitude	Longitude	PoIs
Beijing	39° N–40° N	116° E–117° E	606,306
Shanghai	30.7° N–31.7° N	121° E–122° E	470,676
Guangzhou	23° N–24° N	113° E–114° E	554,114
Shenzhen	22° N–23° N	114° E–115° E	322,677
Tianjin	39° N–40° N	117° E–118° E	212,838

TABLE 2
Overview of Foursquare PoI Datasets

Area	Latitude	Longitude	PoIs
New York City	40.4°–41.0° N	74.3°–73.6° W	453,070
Belgium	49.4°–51.6° N	2.5°–6.5° E	446,354
Singapore	1.1°–1.5° N	103.6°–104.1° E	400,520
Seoul	37.4°–37.8° N	126.7°–127.2° E	329,120

comparison with the existing methods [7], [8]. To further reduce the number of queries, we propose a mix method RRZI_URS, which first picks a small sub-region from the area of interest at random and then samples PoIs within the sub-region using RRZI. Moreover, for map services such as Google maps providing the total number of PoIs within an input search region, we propose a method to improve the accuracy of RRZI by utilizing this meta information. We perform experiments using a variety of real datasets, and show that our methods dramatically reduce the number of queries required to achieve the same estimation accuracy of state-of-the-art methods.

The rest of the paper is organized as follows. The problem is formulated in Section 2. Section 3 presents our methods for sampling PoIs on maps and estimating the aggregate statistics of PoIs within the area of interest. Performance evaluation and real applications on Foursquare, Google, and Baidu maps [9] are presented in Sections 4 and 5. Section 6 summarizes related work. Concluding remarks then follow.

2 Background

In this section, we first formulate the problem, and then conduct an in-depth analysis of several real datasets to show the challenges of solving the problem.

2.1 Objectives

Our aim is to estimate aggregate statistics (e.g., sum, average, and distribution) of PoIs' attributes. Formally, let $\mathbb{A}$ be the area of interest. Denote by $\mathbb{P}$ the set of PoIs within $\mathbb{A}$. For example, $\mathbb{P}$ can be the set of hotels within $\mathbb{A}$. We want to estimate the following statistics over $\mathbb{P}$.

1) *Sum aggregate*. For any function $f : \mathbb{P} \rightarrow \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers, the sum aggregate is defined as $f_s(\mathbb{P}) = \sum_{p \in \mathbb{P}} f(p)$. If $f(p)$ is the number of rooms a hotel p has, then $f_s(\mathbb{P})$ corresponds to the total number of hotel rooms within $\mathbb{A}$. If $f(p)$ is the constant function $f(p) = 1$, then $f_s(\mathbb{P})$ corresponds to $|\mathbb{P}|$, the number of hotels within $\mathbb{A}$.
2) *Average aggregate*. For any function $f : \mathbb{P} \rightarrow \mathbb{R}$, the average aggregate is defined as $f_a(\mathbb{P}) = \frac{1}{|\mathbb{P}|} \sum_{p \in \mathbb{P}} f(p)$. If $f(p)$ is the price per room per night for a hotel p, then $f_a(\mathbb{P})$ corresponds to the average price for hotels within $\mathbb{A}$.
3) *PoI distribution*. Let $L(p)$ be the label of a PoI p specifying a certain property of p. For example, $L(p)$ can be the star rating of a hotel p. Denote the range of PoI labels as $\{l_1, \ldots, l_J\}$. Let $\theta = (\theta_1, \ldots, \theta_J)$ be the distribution of a set of PoIs, where θ_j $(1 \leq j \leq J)$ is the fraction of PoIs with label l_j. Formally, $\theta_j = \frac{1}{|\mathbb{P}|} \sum_{p \in \mathbb{P}} \mathbf{1}(L(p) = l_j), 1 \leq j \leq J$, where $\mathbf{1}(L(p) = l_j)$ is the indicator function that equals one when predicate $L(p) = l_j$ is true, and zero otherwise. If $L(p)$ is the star rating of p, then θ is the star rating distribution of hotels within $\mathbb{A}$.

As alluded before, we focus on designing *sampling* methods to accurately estimate the above statistics from as few queries as possible.

2.2 Datasets

Tables 1 and 2 summarize the public datasets used in this paper. Table 1 shows the statistics of PoI datasets [10] collected from Baidu maps for five areas in China. Each area spans a degree of longitude and a degree of latitude. They cover five popular cities in China: Beijing, Shanghai, Guangzhou, Shenzhen, and Tianjin. Each dataset consists of more than 200 thousand PoIs. Table 2 shows the statistics of four Foursquare PoI datasets [6], which are exhaustively collected from areas in New York City, Belgium, Singapore, and Seoul.

2.3 Challenges

Fig. 1 shows the geographical distribution of PoIs within Beijing. We can see that PoIs are not evenly distributed, but are clustered into sparse and dense areas. The lack of the PoI geographical distribution makes it challenging to accurately estimate PoI statistics, since it is hard to sample PoIs from $\mathbb{A}$ uniformly. Suppose that one divides $\mathbb{A}$ evenly into non-overlapping sub-regions with size: d degrees of longitude $\times$ d degrees of latitude. Then sample a set of sub-regions at random and collect PoIs within sub-regions sampled. Next, we show that this straightforward method cannot be implemented in practice.

Let k be the maximum number of PoIs returned for a query. A region is defined as an inaccessible sub-region when it includes more than k PoIs. Fig. 2 shows the fraction

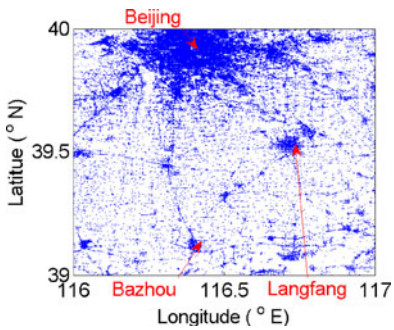


Fig. 1. (Baidu maps) PoI geographical distributions.

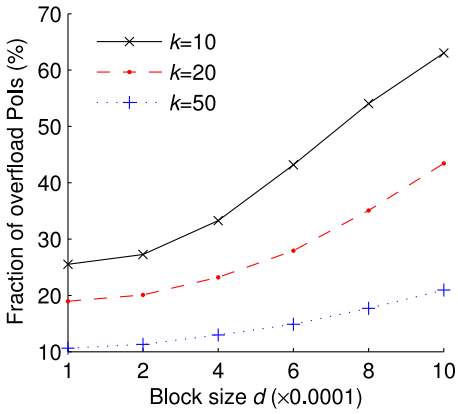


Fig. 2. (Baidu maps) Fractions of inaccessible PoIs.

of inaccessible PoIs, where inaccessible PoIs refer to the PoIs within inaccessible sub-regions. We can see that the number of inaccessible PoIs increases with d. When $d = 0.001$, more than 30 percent of PoIs are inaccessible for $k = 10$ and $k = 20$. For a sampled inaccessible sub-region r, therefore, it might include a large number of PoIs. Then we need to further divide r and explore all PoIs within r, which might require a large number of queries since r might include a lot of PoIs. Otherwise, k PoIs within r are returned by public APIs and they might not be randomly selected from the PoIs within r. This introduces unknown errors into estimates of PoI's statistics. Fig. 2 shows that almost 10 percent of PoIs are within inaccessible regions even when $d = 0.0001$. Thus, a quite small value of d is required to ensure there is no inaccessible sub-region.

However a small d dramatically increases the number of queries required to sample a non-empty sub-region (i.e., a region with at least one PoI). Define ρ as the fraction of non-empty sub-regions among all sub-regions. Then, the hit ratio defined as the probability of sampling a non-empty sub-region is ρ. To sample a non-empty sub-region, on average we need to sample and query $1/\rho$ sub-regions at random. Fig. 3 shows the hit ratios for different d. We can see that ρ increases with d. When $d = 0.0001$, almost 99.8 percent of sub-regions do not include any PoI. Therefore, on average more than 500 queries are required to sample a non-empty sub-region for $d \leq 0.0001$.

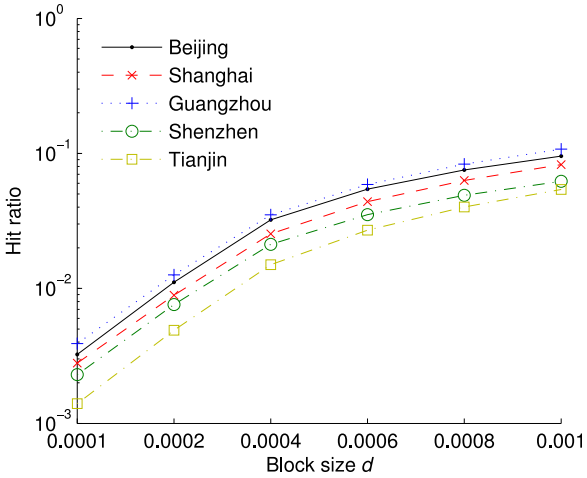


Fig. 3. (Baidu maps) Hitting ratios.

TABLE 3
Table of Notations

$\mathbb{A}$	area of interest
$\mathbb{P}$	set of PoIs within $\mathbb{A}$
k	maximum number of PoIs returned in a response to a query
$\tau(Q, A), Q \subseteq A$	probability of sampling a region Q from A
$n(Q)$	number of PoIs within a region Q
$\chi_0(Q)$, $\chi_1(Q)$	two sub-regions obtained by dividing Q
δ	minimum acceptable latitude and longitude precision of map APIs
L	parameter to control the size of regions sampled by URS
B_L	set of sub-regions obtained by iteratively applying L times region division operations into $\mathbb{A}$
B_L^*	set of non-empty regions in B_L
$P(r)$	set of PoIs within a region r
V	set of fully accessible regions obtained by iteratively applying region division operations into $\mathbb{A}$
m	number of sampled fully accessible regions

Results for Foursquare datasets are similar, which are omitted here. In summary, the above straightforward sampling method is not easy to be implemented, so designing accurate and efficient sampling methods for estimating PoI statistics is a much challenging task.

3 Random Region Zoom-in on Maps

In this section, we present our sampling methods to estimate PoI aggregate statistics defined in Section 2. We first propose a random region zoom-in method to sample PoIs within an area $\mathbb{A}$ of interest, and give our estimators of PoI statistics. To improve the accuracy of RRZI, we then propose a method RRZIC by utilizing the meta information (i.e., the total number of PoIs within an input search region) returned for a query, which is provided by map services such as Google maps. To further reduce the number of queries of RRZI and RRZIC required, we propose mix methods RRZI_URS and RRZIC_URS, which first pick a small sub-region from $\mathbb{A}$ at random and then sample PoIs within the sub-region using RRZI and RRZIC respectively. Finally, we show that RRZIC_URS might exhibit larger errors than RRZIC for estimating PoI statistics. To solve this problem, we propose a method RRZIC_MHWRS to increase the accuracy of RRZIC_URS. For ease of reading, we list notations used throughout the paper in Table 3.

3.1 Random Region Zoom-in

As shown in Fig. 4, RRZI($\mathbb{A}$) works as follows: From the initial region $\mathbb{A}$, RRZI divides the current queried region into two sub-regions without overlapping, and then randomly selects a non-empty sub-region as the next region to zoom in and query. Repeat this procedure recursively until a fully accessible region is finally observed. The reason of why we divide the current region into just two sub-regions is because it minimizes the number of queries required at each step, which will be discussed later. Algorithm 1 shows the pseudo-code of RRZI($\mathbb{A}$). Initially RRZI sets region $Q = \mathbb{A}$.

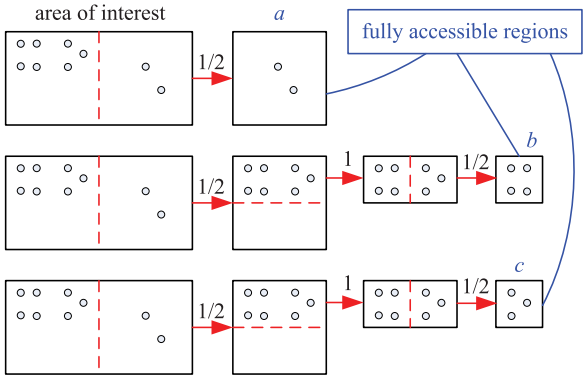


Fig. 4. An example of applying RRZI into the area of interest, where $k = 5$. The number above a red arrow refers to the probability of selecting a sub-region of the current queried region to zoom in.

At each step, RRZI uses a query $searchPoI(Q)$ to explore PoIs within Q, and then determines whether Q is a fully accessible region depending on the value of $|O|$, the number of PoIs returned by $searchPoI(Q)$. When $|O|$ equals k, Q might not be a fully accessible region, and RRZI divides Q into two non-overlapping sub-regions Q_0 and Q_1 to further explore. If Q_0 and Q_1 are both non-empty regions, RRZI randomly selects one as the next region, that is, $Q \leftarrow random(Q_0, Q_1)$. Otherwise, RRZI sets Q as the non-empty region among Q_0 and Q_1. RRZI repeats this procedure until a fully accessible region Q is observed.

Algorithm 1. RRZI($\mathbb{A}$) Pseudo-Code

Input: $\mathbb{A}$
/* Q is a sub-region sampled from $\mathbb{A}$ at random, and τ records the probability of sampling Q from $\mathbb{A}$. */
Output: Q and τ
/* searchPoI(Q) is the set of PoIs returned for querying the region Q */
$Q \leftarrow \mathbb{A}$, $\tau \leftarrow 1$, $l \leftarrow 0$, and $O \leftarrow$ searchPoI(Q);
while $|O| = k$ **do**
  /* $\chi_0(Q)$ and $\chi_1(Q)$ are the two sub-regions of Q defined as (1) and (2) */
  $Q_0 \leftarrow \chi_0(Q)$ and $Q_1 \leftarrow \chi_1(Q)$;
  /* If O includes no PoI in the region Q_0/Q_1, then emptyRegion(Q_0, Q_1, O) returns 0/1. Otherwise, both Q_0 and Q_1 are non-empty, and emptyRegion(Q_0, Q_1, O) returns -1 */
  $i \leftarrow$ emptyRegion(Q_0, Q_1, O);
  if $i \neq -1$ **then**
    $O' \leftarrow$ searchPoI(Q_i);
    if $|O'| = 0$ **then**
      $Q \leftarrow Q_{1-i}$;
  else
      /* random(Q_0, Q_1) returns Q_0 and Q_1 at random */
      $Q \leftarrow$ random(Q_0, Q_1);
      $O \leftarrow O'$ and $\tau \leftarrow \tau/2$;
  end
 else
      $Q \leftarrow$ random(Q_0, Q_1);
      $O \leftarrow$ searchPoI(Q) and $\tau \leftarrow \tau/2$;
  end
end

Next, we answer the following three *critical* questions.

- *How to divide Q into two non-overlapping regions Q_0 and Q_1?* For simplicity, we denote $Q = [(x_{SW}, y_{SW}), (x_{NE}, y_{NE})]$ as a quadrangle region with south-west corner (x_{SW}, y_{SW}) (latitude and longitude pair) and north-east corner (x_{NE}, y_{NE}). Let δ be the minimum acceptable latitude and longitude precision of map APIs. Let $\beta(x_{SW}, x_{NE})$ be a function whose values are positive real numbers. We define functions $\chi_0(Q)$ and $\chi_1(Q)$ as follows: If $|x_{SW} - x_{NE}| \geq \beta(x_{SW}, x_{NE})\,|y_{SW} - y_{NE}|$, then

$$\begin{cases} \chi_0(Q) = [(x_{SW}, y_{SW}), (\lceil \frac{x_{SW}+x_{NE}}{2\delta} \rceil \delta - \delta, y_{NE})] \\ \chi_1(Q) = [(\lceil \frac{x_{SW}+x_{NE}}{2\delta} \rceil \delta, y_{SW}), (x_{NE}, y_{NE})]. \end{cases} \quad (1)$$

  Otherwise,

$$\begin{cases} \chi_0(Q) = [(x_{SW}, y_{SW}), (x_{NE}, \lceil \frac{y_{SW}+y_{NE}}{2\delta} \rceil \delta - \delta)] \\ \chi_1(Q) = [(x_{SW}, \lceil \frac{y_{SW}+y_{NE}}{2\delta} \rceil \delta), (x_{NE}, y_{NE})]. \end{cases} \quad (2)$$

  Using functions χ_0 and χ_1 we divide Q into two sub-regions $\chi_0(Q)$ and $\chi_1(Q)$ without overlapping, which almost have the same size. $\beta(x_{SW}, x_{NE})$ is used to control the shape of a sub-region finally sampled. For instance, we let $\beta(x_{SW}, x_{NE}) = 1/\cos(0.5(x_{SW} + x_{NE}))$, and then the shape of a fully accessible region sampled by RRZI approaches to a square, since the physical distance between two points on the earth with the same latitude x but different longitudes y and $y + d$ is $111d \times \cos x$ kilometers, and the distance between two points with the same longitudes y but different latitudes x and $x + d$ is $111d$.
- *How to determine whether $\chi_0(Q)$ and $\chi_1(Q)$ are empty regions or not using a minimum number of queries?* Let O be the set of PoIs within the region Q observed by previous queries. If O includes PoIs within both $\chi_0(Q)$ and $\chi_1(Q)$, then neither of $\chi_0(Q)$ and $\chi_1(Q)$ is empty. Otherwise, we query the sub-region with no PoIs in O to determine whether it is a truly empty region. Therefore RRZI needs at most one query at each step.
- *Does RRZI sample PoIs uniformly? If not, how to remove the sampling bias?* Fig. 4 shows an example of applying RRZI into $\mathbb{A}$, where $k = 5$. We can see that there exist three fully accessible regions a, b, and c, which could be observed and sampled by RRZI. The probabilities of sampling a, b, and c are $1/2$, $1/4$, and $1/4$ respectively. The sampling bias might introduce large errors into the measurement of PoI statistics. To solve this problem, we use a counter τ to record the probability of sampling a region from $\mathbb{A}$, which is used to correct the sampling bias later. τ is initialized with 1, and updated as follows: At each step, we set $\tau = \tau/2$ if both $\chi_0(Q)$ and $\chi_1(Q)$ are non-empty, otherwise τ keeps unchanged. Finally τ records the probability of sampling a fully accessible sub-region from $\mathbb{A}$.

At last, we analyze the query cost of RRZI and present our estimators of PoI aggregate statistics under study. We can easily find that the number of queries required to sample a fully accessible region is smaller than

$$H_{max} = \log(L_x/\delta) + \log(L_y/\delta), \tag{3}$$

where L_x and L_y are the length and width of the region $\mathbb{A}$. For instance, H_{max} equals 60 when $\delta = 10^{-6}$ and $\mathbb{A}$ is a region with size: one degree of longitude $\times$ one degree of latitude. Experimental results in a later section show that the associated average number of queries required is about 13, which is much smaller than H_{max}.

Denote V as the set of fully accessible regions, which can be observed by RRZI. Define $\tau(Q, \mathbb{A})$ as the probability of sampling a fully accessible region Q from $\mathbb{A}$. Then we have $\sum_{r\in V} \tau(r, \mathbb{A}) = 1$. For the example in Fig. 4, there exist three fully accessible regions a, b, and c which can be sampled by RRZI, thus $V = \{a, b, c\}$. RRZI($\mathbb{A}$) returns a region among a, b, and c according to probabilities $\tau(a, \mathbb{A}) = 1/2$, $\tau(b, \mathbb{A}) = 1/4$, and $\tau(c, \mathbb{A}) = 1/4$ respectively. Let $P(Q)$ be the set of PoIs within Q. Given a set of fully accessible regions r_i $(1 \le i \le m)$ sampled by calling the function RRZI($\mathbb{A}$) m times, we collect PoIs within r_i, and estimate the sum aggregate $f_s(\mathbb{P})$ as

$$\tilde{f}_s(\mathbb{P}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{f(p)}{\tau(r_i, \mathbb{A})}.$$

Note that there might exist duplicated regions among r_i.

Theorem 1. *$\tilde{f}_s(\mathbb{P})$ is an* **consistent** *estimator of $f_s(\mathbb{P})$, that is, $E[\tilde{f}_s(\mathbb{P})] = f_s(\mathbb{P})$.*

Proof. $r_1, \ldots, r_m$ are sampled independently, thus

$$E[\tilde{f}_s(\mathbb{P})] = \sum_{r\in V}\sum_{p\in P(r)}\frac{f(p)}{\tau(r, \mathbb{A})} \times \tau(r, \mathbb{A}) = f_s(\mathbb{P}).$$

The variance of $\tilde{f}_s(\mathbb{P})$ is

$$Var(\tilde{f}_s(\mathbb{P})) = \frac{1}{m}\left(\sum_{r\in V}\frac{\left(\sum_{p\in P(r)} f(p)\right)^2}{\tau(r, \mathbb{A})} - f_s^2(\mathbb{P})\right).$$

We omit the proof due to the limited space. $\left(\sum_{r\in V}\frac{\left(\sum_{p\in P(r)} f(p)\right)^2}{\tau(r,\mathbb{A})} - f_s^2(\mathbb{P})\right)$ is constant value and independent with samples, therefore $Var(\tilde{f}_s(\mathbb{P}))$ is inversely proportional to m, i.e., the number of sampled sub-regions. It indicates that the estimator $\tilde{f}_s(\mathbb{P})$ is *consistent* because $Var(\tilde{f}_s(\mathbb{P}))$ goes to zero when $m \to \infty$. □

Remark 1. In practice it is important to bound an estimate's error. However it requires the knowledge of all PoIs' information to compute $Var(\tilde{f}_s(\mathbb{P}))$. To solve this problem, we define

$$\tilde{\sigma}_s^2 = \frac{1}{m}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{\left(\sum_{p\in P(r_i)} f(p)\right)^2}{\tau^2(r_i, \mathbb{A})} - \tilde{f}_s^2(\mathbb{P})\right). \tag{4}$$

Since $E[\tilde{\sigma}_s^2] = Var(\tilde{f}_s(\mathbb{P}))$, we compute the 95 percent confidence interval of $\tilde{f}_s(\mathbb{P})$ as $[\tilde{f}_s(\mathbb{P}) - 2\tilde{\sigma}_s, \tilde{f}_s(\mathbb{P}) + 2\tilde{\sigma}_s]$ based on the empirical rule.

Similarly, we estimate the PoI distribution $\theta = (\theta_1, \ldots, \theta_J)$ as follows

$$\tilde{\theta}_j = \frac{1}{\tilde{H}}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{\mathbf{1}(L(p) = l_j)}{\tau(r_i, \mathbb{A})}, \qquad 1 \le j \le J,$$

where $\tilde{H} = \sum_{i=1}^{m}\frac{n(r_i)}{\tau(r_i,\mathbb{A})}$. Let $n_j(\mathbb{A})$ be the number of PoIs with label l_j. We can easily find that $\tilde{n}(\mathbb{A}) = \frac{\tilde{H}}{m}$ and $\tilde{n}_j(\mathbb{A}) = \frac{\tilde{H}\tilde{\theta}_j}{m}$ are consistent estimators of $n(\mathbb{A})$ and $n_j(\mathbb{A})$ respectively, and their standard errors $\tilde{\sigma}_s(\tilde{n}(\mathbb{A}))$ and $\tilde{\sigma}_s(\tilde{n}_j(\mathbb{A}))$ can be computed similar to (4). Finally, we compute the 95 percent confidence interval of $\tilde{\theta}_j$ as $[\frac{\tilde{n}_j(\mathbb{A})-2\tilde{\sigma}_s(\tilde{n}_j(\mathbb{A}))}{\tilde{n}(\mathbb{A})+2\tilde{\sigma}_s(\tilde{n}(\mathbb{A}))}, \frac{\tilde{n}_j(\mathbb{A})+2\tilde{\sigma}_s(\tilde{n}_j(\mathbb{A}))}{\tilde{n}(\mathbb{A})-2\tilde{\sigma}_s(\tilde{n}(\mathbb{A}))}]$. In this section, we do not present our method for estimating the average aggregate $f_a(\mathbb{P})$. $f_a(\mathbb{P})$ is easily computed based on our estimates of sum aggregates since $f_a(\mathbb{P}) = f_s(\mathbb{P})/n(\mathbb{A})$. Note that $n(\mathbb{A})$ is obtained by estimating the sum aggregate with the function $f(p) = 1$ when it is not known in advance.

Remark 2. Clearly, PoIs sampled by RRZI are not independent. To eliminate estimation errors introduced by the correlation of samples, we can pick one and only one PoI p_i from a sampled fully accessible region r_i at random, and discards the other PoIs in r_i. We can find that each PoI in r_i is sampled with the same probability $\tau(r_i, \mathbb{A})/n(r_i)$, and then sampled PoIs are independent. We denote this method as RRZI*. RRZI* estimates $f_s(\mathbb{P})$ and $\theta = (\theta_1, \ldots, \theta_J)$ as

$$\tilde{f}_s^*(\mathbb{P}) = \frac{1}{m}\sum_{i=1}^{m}\frac{n(r_i)f(p_i)}{\tau(r_i, \mathbb{A})}.$$

$$\tilde{\theta}_j^* = \frac{1}{\tilde{H}}\sum_{i=1}^{m}\frac{n(r_i)\mathbf{1}(L(p_i) = l_j)}{\tau(r_i, \mathbb{A})}, \qquad 1 \le j \le J.$$

Later, our experimental results show that $\tilde{f}_s^*(\mathbb{P})$ and $\tilde{\theta}_j^*$ exhibit larger errors than $\tilde{f}_s(\mathbb{P})$ and $\tilde{\theta}_j$, this may because RRZI* discards most PoIs sampled by RRZI.

3.2 Random Region Zoom-in with Count Information

In this section we propose a method, named random region zoom-in with count information (RRZIC), to further improve the accuracy of RRZI for map services such as Google maps, where results from a query include the number of PoIs within the input search region. Compared to RRZI, RRZIC tends to sample PoIs uniformly, giving us smaller estimation errors for PoIs statistics. The pseudo-code RRZIC($\mathbb{A}$) is shown as Algorithm 2. Initially we set $Q = \mathbb{A}$. Denote z as the number of PoIs within the current queried region Q. Let z_0 and z_1 be the number of PoIs within the two sub-regions $\chi_0(Q)$ and $\chi_1(Q)$ of Q respectively. If $z > k$, Q is not a fully accessible region, and RRZIC queries $\chi_0(Q)$ to obtain z_0. With probability z_0/z, RRZIC then selects $\chi_0(Q)$ to further explore. That is, set $Q = \chi_0(Q)$ and $z = z_0$. Or, RRZIC sets $Q = \chi_1(Q)$ and $z = z_1$ with probability z_1/z. RRZIC repeats this procedure until a fully accessible Q is observed. We can easily find that RRZIC samples Q from $\mathbb{A}$ with probability $\tau(Q, \mathbb{A}) = n(Q)/n(\mathbb{A})$, where $n(Q)$ is the number of PoIs within Q.

Given m fully accessible regions r_i $(1 \le i \le m)$ sampled by RRZIC, we estimate $f_s(\mathbb{P})$ as

$$\bar{f}_s(\mathbb{P}) = \frac{1}{m}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{f(p)n(\mathbb{A})}{n(r_i)}, \tag{5}$$

and estimate $\theta = (\theta_1, \ldots, \theta_J)$ as

$$\bar{\theta}_j = \frac{1}{m}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{\mathbf{1}(L(p)=l_j)}{n(r_i)}, \qquad 1 \le j \le J. \tag{6}$$

Their variances are

$$Var(\bar{f}_s(\mathbb{P})) = \frac{1}{m}\left(\sum_{r\in V}\frac{\left(\sum_{p\in P(r)} f(p)\right)^2 n(\mathbb{A})}{n(r)} - f_s^2(\mathbb{P})\right)$$

$$Var(\bar{\theta}_j) = \frac{1}{m}\left(\sum_{r\in V}\frac{\left(\sum_{p\in P(r)} \mathbf{1}(L(p)=l_j)\right)^2}{n(r)n(\mathbb{A})} - \theta_j^2\right).$$

Similar to $\tilde{f}_s(\mathbb{P})$, we can easily prove that $\bar{f}_s(\mathbb{P})$ and $\bar{\theta}_j$ are *consistent* estimators of $f_s(\mathbb{P})$ and θ_j, i.e, $E[\bar{f}_s(\mathbb{P})] = f_s(\mathbb{P})$, $\lim_{m\to\infty} Var(\bar{f}_s(\mathbb{P})) = 0$, $E[\bar{\theta}_j] = \theta_j$, and $\lim_{m\to\infty} Var(\bar{\theta}_j) = 0$. Moreover, we estimate $Var(\bar{f}_s(\mathbb{P})) \approx \bar{\sigma}_s^2$ and $Var(\bar{\theta}_j) \approx \bar{\sigma}_{\theta_j}^2$, where $\bar{\sigma}_s^2$ and $\bar{\sigma}_{\theta_j}^2$ are defined as

$$\bar{\sigma}_s^2 = \frac{1}{m}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{\left(\sum_{p\in P(r_i)} f(p)\right)^2 n(\mathbb{A})^2}{n(r_i)^2} - \bar{f}_s^2(\mathbb{P})\right).$$

$$\bar{\sigma}_{\theta_j}^2 = \frac{1}{m}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{\left(\sum_{p\in P(r_i)} \mathbf{1}(L(p)=l_j)\right)^2}{n(r_i)^2} - \bar{\theta}_j^2\right).$$

Algorithm 2. RRZIC($\mathbb{A}$) Pseudo-Code

Input: $\mathbb{A}$
/* Q is a sub-region sampled from $\mathbb{A}$ at random, and τ records the probability of sampling Q from $\mathbb{A}$. */
Output: Q and τ
/* countPoI(Q) returns the number of PoIs in Q. */
$Q \leftarrow \mathbb{A}$, $\tau \leftarrow 1$, and $z \leftarrow$ countPoI(Q);
/* k is the maximum number of PoIs returned in a response to a query. */
while $z > k$ **do**
  /* $\chi_0(Q)$ and $\chi_1(Q)$ are the two sub-regions of Q defined as (1) and (2). */
  $Q_0 \leftarrow \chi_0(Q)$ and $Q_1 \leftarrow \chi_1(Q)$;
  /* z_0 and z_1 are the numbers of PoIs within the regions Q_0 and Q_1 respectively. */
  $z_0 \leftarrow$ countPoI(Q_0) and $z_1 = z - z_0$;
  /* $U(0,1)$ is a random sample from $(0,1)$. */
  $u \leftarrow U(0,1)$;
  if $u < z_0/z$ **then**
    $Q \leftarrow Q_0$, $\tau \leftarrow \tau \times z_0/z$, and $z \leftarrow z_0$;
  else
    $Q \leftarrow Q_1$, $\tau \leftarrow \tau \times z_1/z$, and $z \leftarrow z_1$;
  end
end

3.3 Random Region Zoom-in Combined with Uniform Region Sampling

Public map APIs might impose a limit on the size of input regions. For example, Foursquare returns an error message "Your geographic boundary is too big. Please search a smaller area." for a query specified with an input quadrangle region with size: 3 degrees of longitude $\times$ 3 degrees of latitude, although it does not literally state any limit on the input quadrangle region. Thus, RRZI and RRZIC cannot be directly applied to sample PoIs from a large area on Foursquare. To solve this problem, we propose mix methods, which first pick a small sub-region from $\mathbb{A}$ at random and then sample PoIs within the sub-region using RRZI and RRZIC. Moreover, we show that our mix methods also reduce the number of queries required to sample a fully accessible region in comparison with RRZI and RRZIC.

We first introduce a uniform region sampling (URS) method, which is used to sample sub-regions from $\mathbb{A}$ uniformly. Let L be a parameter to control the size of sub-regions sampled by URS. Denote B_L as the set of sub-regions of $\mathbb{A}$ obtained by iteratively applying L times region division operations defined as (1) and (2) into $\mathbb{A}$. Formally,

$$B_L = \{\chi_{i_1}(\chi_{i_2}(\ldots(\chi_{i_L}(\mathbb{A})))) : i_1, i_2, \ldots, i_L \in \{0,1\}\}.$$

B_L consists of 2^L regions. That is, $|B_L| = 2^L$. Regions in B_L are nearly 2^L times smaller than $\mathbb{A}$. We assume that they are small enough to be used as input regions for public APIs. Let B_L^* be the set of non-empty regions in B_L. To sample a region from B_L^* uniformly, URS repeats to sample regions from B_L at random until a non-empty region is observed.

Algorithm 3. RRZI_URS/RRZIC_URS Pseudo-Code

Input: $\mathbb{A}$ and L
/* r_i, $1 \le i \le m$, are sub-regions sampled from $\mathbb{A}$ randomly, and $\tau(r_i, b_i)$ records the probability of sampling r_i from a region b_i, where b_i is sampled from B_L^* at random. */
Output: $r_1, \ldots, r_m$ and $\tau(r_1, b_1), \ldots, \tau(r_m, b_m)$.
$i \leftarrow 1$;
while $i \le m$ **do**
  /* URS($\mathbb{A}$) returns a region sampled from the set B_L^* at random. */
  $b_i \leftarrow$ URS($\mathbb{A}$);
  /* The following statement is used for RRZI_URS. For RRZIC_URS, it should be $[r_i, \tau(r_i, b_i)] \leftarrow$ RRZIC(b_i) */
  $[r_i, \tau(r_i, b_i)] \leftarrow$ RRZI(b_i) and $i \leftarrow i + 1$;
end

Our mix methods RRZI_URS and RRZIC_URS are modifications of RRZI and RRZIC. Algorithm 3 shows their pseudo-codes. RRZI_URS and RRZIC_URS first randomly select a non-empty region b from B_L using URS, and then sample a fully accessible region from b using RRZI(b) and RRZIC(b) respectively. Let $|\hat{B}_L^*|$ be an estimate of $|B_L^*|$, which can be easily estimated based on the hit ratio of sampling a non-empty region from B_L using URS. Given m fully accessible regions r_i ($1 \le i \le m$) sampled by RRZI_URS or RRZIC_URS, we estimate $f_s(\mathbb{P})$ and $\theta = (\theta_1, \ldots, \theta_J)$ as

- **Estimators for RRZI_URS:**

$$\acute{f}_s(\mathbb{P}) = \frac{|\hat{B}_L^*|}{m}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{f(p)}{\tau(r_i,b_i)};$$

$$\acute{\theta}_j = \frac{1}{\acute{H}}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{\mathbf{1}(L(p)=l_j)}{\tau(r_i,b_i)}, \qquad 1\le j\le J,$$

where $\acute{H}=\sum_{i=1}^{m}\frac{n(r_i)}{\tau(r_i,b_i)}$.

- **Estimators for RRZIC_URS:**

$$\grave{f}_s(\mathbb{P}) = \frac{|\hat{B}_L^*|}{m}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{f(p)n(b_i)}{n(r_i)};$$

$$\grave{\theta}_j = \frac{1}{\grave{H}}\sum_{i=1}^{m}\sum_{p\in P(r_i)}\frac{\mathbf{1}(L(p)=l_j)n(b_i)}{n(r_i)}, \quad 1\le j\le J,$$

where $\grave{H}=\sum_{i=1}^{m}n(b_i)$.

Compared to RRZIC, although RRZIC_URS reduces the number of queries required to sample a fully accessible region, its estimates $\grave{f}_s(\mathbb{P})$ and $\grave{\theta}_j$ might exhibit larger errors, since PoI dense regions might be under-sampled by URS. Next, we propose a Metropolis-Hastings based weighted region sampling (MHWRS) to address this problem, which samples non-empty regions from B_L according to the probability distribution $\pi=(\pi_b=n(b)/n(\mathbb{A}):b\in B_L^*)$. Compared to URS, MHWRS draws more samples from dense regions to reduce the variance of PoI statistic estimates. URS can be modeled as a Markov chain with transition matrix $P=[P_{b,b^*}]$, $b,b^*\in B_L^*$, where $P_{b,b^*}=\frac{1}{|B_L^*|}$ is defined as the probability that a region b^* is selected as the next sampled region given that the current region is b. To generate a sequence of random samples from a desired stationary distribution $\pi=(\pi_b:b\in B_L^*)$, the Metropolis-Hastings technique [11], [12], [13] is a Markov chain Monte Carlo method based on modifying the transition matrix of URS as

$$P^{\bigstar}_{b,b^*}=\begin{cases} P_{b,b^*}\min\left(\frac{\pi_{b^*}P_{b^*,b}}{\pi_b P_{b,b^*}},1\right) & \text{if } b^*\neq b,\\ 1-\sum_{b'\neq b}P^{\bigstar}_{b,b'} & \text{if } b^*=b.\end{cases}$$

It provides a way to alter the next region selection to produce any desired stationary distribution π. MHWRS with target distribution $\pi=(\pi_b=n(b)/n(\mathbb{A}):b\in B_L^*)$ works as follows: At each step, MHWRS selects a region b^* using URS and then accepts the move with probability $\min\left(\frac{n(b^*)}{n(b)},1\right)$. Otherwise, MHWRS remains at the current region b. The pseudo-code of our mix method RRZIC_MHWRS is shown as Algorithm 4. It first randomly samples a non-empty region b from B_L using MHWRS and then samples a fully accessible region from b using $\mathrm{RRZIC}(b)$. We can easily find that the probability of RRZIC_MHWRS sampling a fully accessible region r converges to $n(r)/n(\mathbb{A})$, which is the same as RRZIC. Given m fully accessible regions r_i $(1\le i\le m)$ sampled by RRZIC_MHWRS, we estimate $f_s(\mathbb{P})$ and $\theta=(\theta_1,\ldots,\theta_J)$ using Eqs. (5) and (6).

Finally, we analyze the query costs of the above mixing methods. To sample a non-empty region from B_L at random, on average URS needs to sample and query $|B_L|/|B_L^*|$ regions from B_L, but RRZI and RRZIC require L queries. Thus, the mixing methods are more efficient than RRZI and RRZIC when $|B_L|/|B_L^*|<L$, which is true for small values of L.

TABLE 4
Comparison of Our Methods

Methods	API with PoI count	Nonparametric
RRZI	Not required	Yes
RRZIC	Required	Yes
RRZI_URS	Not required	No
RRZIC_URS	Required	No
RRZIC_MHWRS	Not required	No

Algorithm 4. RRZIC_MHWRS Pseudo-Code

Input: $\mathbb{A}$
/* $r_1,\ldots,r_m$ are sub-regions sampled from $\mathbb{A}$ at random, and $\tau(r_i,b_i)$ records the probability of sampling r_i from a region b_i, where b_i is sampled from the set B_L^* at random. */
Output: $r_1,\ldots,r_m$ and $\tau(r_1,b_1),\ldots,\tau(r_m,b_m)$
/* $\mathrm{URS}(\mathbb{A})$ returns a region sampled from the set B_L^* at random. $\mathrm{countPoI}(b)$ returns the number of PoIs in b. */
$b\leftarrow\mathrm{URS}(\mathbb{A})$, $z\leftarrow\mathrm{countPoI}(b)$, and $i\leftarrow 1$;
while $i\le m$ **do**
  $b_i\leftarrow b$ and $[r_i,\tau(r_i,b_i)]\leftarrow\mathrm{RRZIC}(b_i)$;
  $b^*\leftarrow\mathrm{URS}(\mathbb{A})$ and $z^*\leftarrow\mathrm{countPoI}(b^*)$;
  /* $U(0,1)$ is a random sample from $(0,1)$. */
  $u\leftarrow U(0,1)$ and $i\leftarrow i+1$;
  if $u\le\min\{z^*/z,1\}$ **then**
    $b\leftarrow b^*$ and $z\leftarrow z^*$;
  end
end

3.4 Comparison

Table 4 compares our methods. In summarize, we can see

1) RRZI and RRZIC are nonparametric, and the mix methods (i.e., RRZI_URS, RRZIC_URS, and RRZIC_MHWRS) require a pilot study to set the parameter L. Compared to RRZI and RRZIC, the mix methods sample a fully accessible region with less queries.
2) RRZIC, RRZIC_URS, and RRZIC_MHWRS require an API whose results from a query include the number of PoIs within the input search region. These methods utilize the meta information returned for a query to reduce estimation errors.

We expect that the most efficient method is RRZIC_MHWRS when there exists a publicly available API with meta information (i.e., the total number of PoIs within an input search region) returned for a query, and RRZI_URS otherwise, which is validated by our experiments later.

4 Data Evaluation

In this section, we conduct experiments on real world datasets listed in Tables 1 and 2 to evaluate the performance of our methods for estimating PoI aggregate statistics.

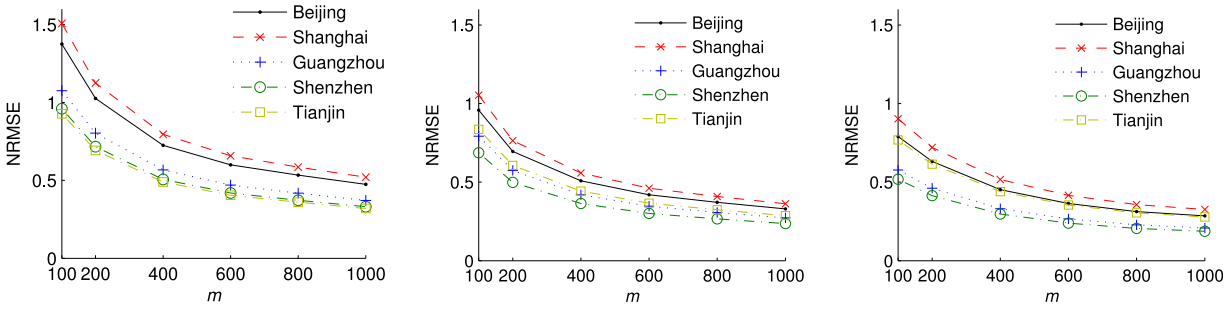


Fig. 5. (Baidu maps) Compared NRMSEs of estimates of $n(\mathbb{A})$ for RRZI with different m, where m is the number of fully accessible regions sampled.

4.1 Results of Estimating $n(\mathbb{A})$

We evaluate the performance of our methods RRZI and RRZI_URS for estimating $n(\mathbb{A})$, the total number of PoIs within the area of interest $\mathbb{A}$, which is an important statistic studied in [7], [8]. In this section, our sampling methods using the PoI count information (i.e., RRZIC, RRZIC_URS, and RRZIC_MHWRS) are not studied, since we assume that map services do not provide a public API where results from a query include the number of PoIs within an input search region (otherwise $n(\mathbb{A})$ can be obtained in a direct manner). We define the normalized root mean square error (NRMSE), that is $\mathrm{NRMSE}(n(\hat{\mathbb{A}})) = \sqrt{E[(\hat{n}(\mathbb{A}) - n(\mathbb{A}))^2]}/n(\mathbb{A})$, be a metric to measure the relative error of the estimate $\hat{n}(\mathbb{A})$ with respect to its true value $n(\mathbb{A})$. In the following experiments, we average the estimates and calculate their NRMSEs over 10,000 runs. Fig. 5 shows the NRMSEs of estimates of $n(\mathbb{A})$ for RRZI with different m, the number of fully accessible regions sampled. We can see that the NRMSEs decrease as m increases, and are roughly proportional to $1/\sqrt{m}$. Meanwhile, RRZI exhibits smaller errors for larger k, the maximum number of PoIs returned in a response to a query. We set $k = 20$ in the following experiments. Fig. 6 shows the average number of queries required to sample a fully accessible region for RRZI_URS with different L, where the y-axis is in log scale. We can see that the query cost of RRZI_URS first decreases with L and then increases with L. It is because URS needs only one or two queries to sample a non-empty sub-region for small and medium L, but a large number of queries is required for large L, which is shown in Fig. 7. In practice, we can conduct a pilot study to estimate the optimal value of L. Fig. 8 shows the compared NRMSEs of estimates of $n(\mathbb{A})$ for RRZI_URS with different L under the same number of queries (10,000 queries). We can see that the NRMSEs first decrease with L and then increase with L. RRZI_URS with $L = 10$, $L = 15$, and $L = 20$ are almost two times more accurate than RRZI_URS with $L = 0$, which is equivalent to RRZI. It indicates that RRZI_URS requires nearly four times less queries than RRZI to achieve the same estimation accuracy, since the NRMSEs are roughly proportional to $1/\sqrt{m}$, which is observed in Fig. 5.

We compare our methods with the state-of-the-art methods *nearest-neighbor search* (NNS) [7] and *random region sampling* (RRS) [8]. Fig. 9 shows the average number of queries required to obtain an estimate of $n(\mathbb{A})$ with NRMSE less than 0.1 for RRZI and RRZI_URS in comparison with NNS and RRS, where we set $L = 15$ and $k = 20$. Here we do not study the performances of *RRZIC*, *RRZIC_URS*, and *RRZIC_MHWRS*, because these three methods assume $n(\mathbb{A})$ can be easily obtained by a query. We can see that RRZI_URS requires eight and six times less queries than RRS and NNS respectively. Here the results of NNS are obtained based on the assumption that a NNS API is supported by map service providers. When map services such as Foursquare do not provide such a NNS API, the PoI closest to a randomly sampled point (x, y) can be obtained by applying a binary search to find a radius r, which ensures at least one and at most k PoIs are within the circle with center (x, y) and radius r. Thus, on average $\log(W/\delta)$ queries are required to get the PoI closest to (x, y), where δ be the minimum acceptable precision for public APIs and W is the

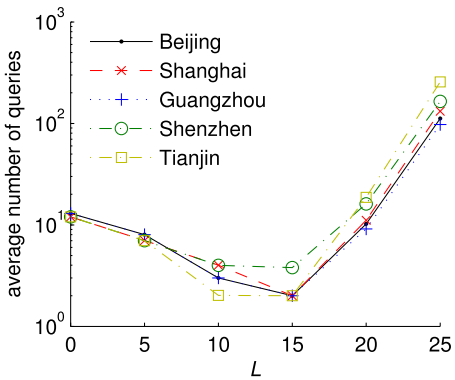


Fig. 6. (Baidu maps) Average number of queries required to sample a fully accessible region for RRZI_URS.

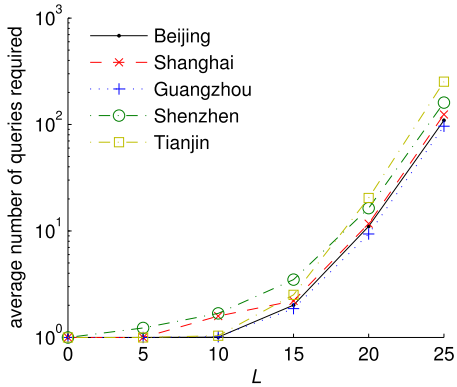


Fig. 7. (Baidu maps) Average number of queries required to sample a non-empty sub-region for URS with different L.

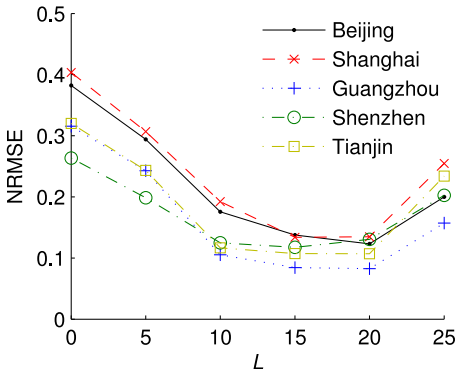


Fig. 8. (Baidu maps) Compared NRMSEs of estimates of $n(\mathbb{A})$ for RRZI_URS with different L under 10,000 queries.

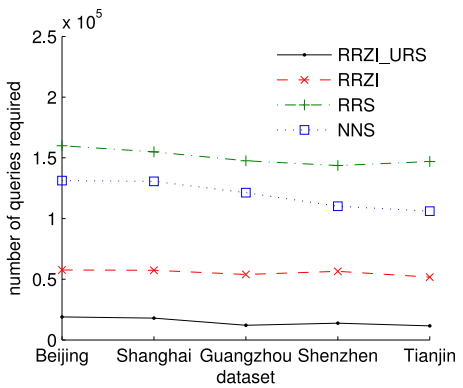


Fig. 9. (Baidu maps) The number of queries required to obtain an estimate of $n(\mathbb{A})$ with NRMSE less than 0.1 for RRZI_URS and RRZI in comparison with the state-of-the-art methods NNS [7] and RRS [8].

TABLE 5
(Baidu Maps) Average Number of Queries Required for Sampling a PoI p and Determining $D(p)$ When the NNS API is Not Available

Area	$k = 10$	$k = 20$	$k = 50$
Beijing	1,196	1,079	995
Shanghai	1,176	1,076	982
Guangzhou	1,172	1,043	969
Shenzhen	1,154	1,038	931
Tianjin	1,106	1,034	909

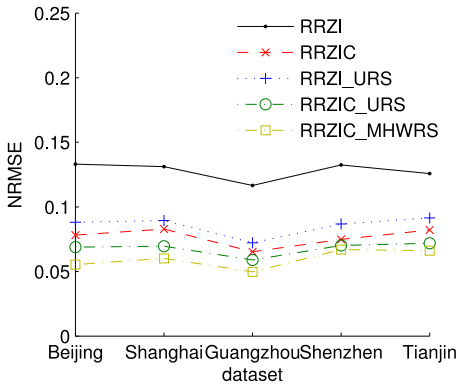
diameter of $\mathbb{A}$. Let $D(p)$ be the set of points in the plane $\mathbb{A}$ that are closer to p than the other PoIs, which is used to correct the sampling bias of the NNS method. Table 5 shows that on average a thousand queries are required to sample a PoI p and determine $D(p)$, which indicates that the NNS method requires many more queries than our methods to achieve the same estimation accuracy when the NNS API is not available.

4.2 Results of Estimating Average and Distribution Statistics

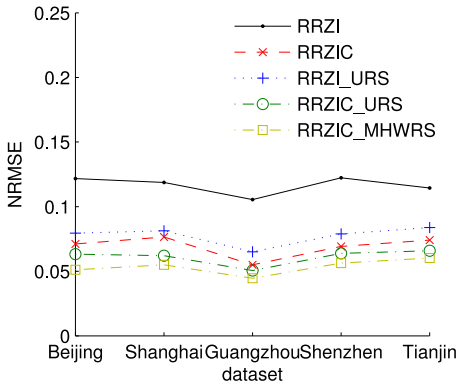
In this section, we conduct experiments to evaluate the performance of our methods for estimating PoIs' average and distribution statistics. For the Baidu PoI datasets we used, PoIs are classified into different types such as restaurant, hotel, and shopping. The numbers of restaurant-tpye PoIs are 75,255, 36,417, 24,353, 16,025, and 10,032 for datasets Beijing, Shanghai, Guangzhou, Shenzhen, and Tianjin

TABLE 6
Average Statistics of Foursquare Datasets

Dataset	Average statistics (per PoI)		
	# tips	# users	# check-ins
New York City	1.23	63.2	151
Belgium	0.27	13.8	47
Singapore	0.55	37.5	102
Seoul	0.46	16.1	37



(a) CDS_UNI

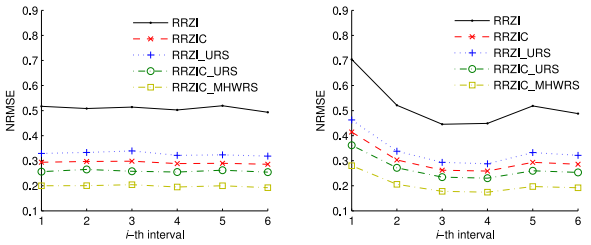


(b) CDS_NOR

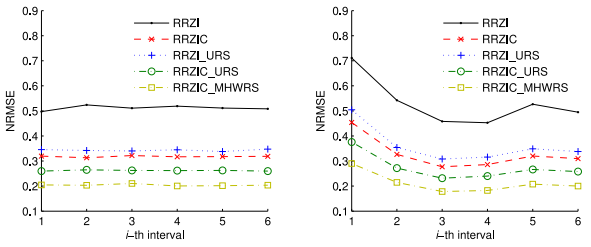
Fig. 10. (Baidu maps) Compared NRMSEs of estimates of restaurant PoIs' average cost for different methods under $1,000$ search queries.

respectively. We use these restaurant-type PoIs to generate benchmark datasets for our following experiments. We manually generate a cost for each restaurant-type PoI using two different cost distribution schemes CDS_UNI and CDS_NOR. For CDS_UNI, the cost of a PoI is uniformly selected from the range (0, 300) at random. For CDS_NOR, the cost of a PoI is a positive number randomly selected from (0, $+\infty$) according to a normal distribution with mean 150 and stand deviation 100. We also conduct experiments on Foursquare datasets. Table 6 and Fig. 12 show the real values of the associated average statistics (i.e., the average numbers of check-ins, users, and tips) and PoI distributions (i.e., the distributions of PoIs by the numbers of check-ins, users, and tips), which are of interest.

Figs. 10 and 11 show our experimental results based on Baidu PoI datasets. Fig. 10 shows the results for estimating the average cost of restaurant-type PoIs for different methods under 1,000 queries, where we set $L = 9$ for our mix methods RRZI_URS, RRZIC_URS, and RRZIC_MHWRS.



(a) Beijing, $\theta^{(\mathrm{UNI})}$ (b) Beijing, $\theta^{(\mathrm{NOR})}$



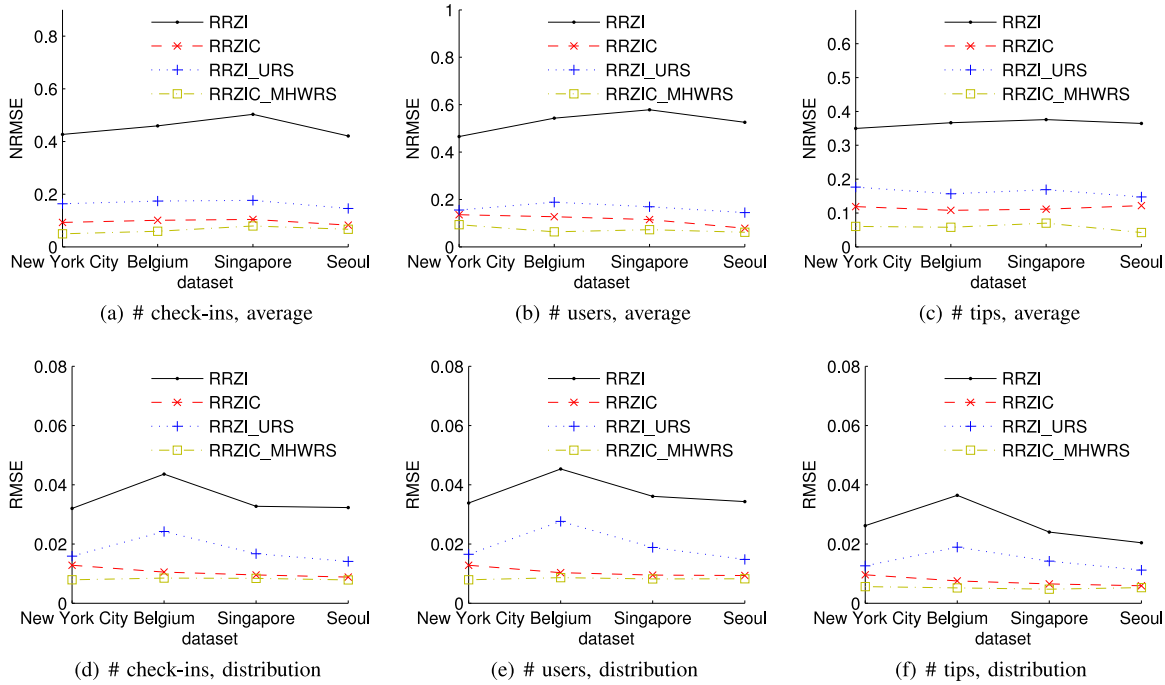
(c) Shanghai, $\theta^{(\mathrm{UNI})}$ (d) Shanghai, $\theta^{(\mathrm{NOR})}$

Fig. 11. (Baidu maps) Compared NRMSEs of estimates of restaurant PoIs' cost distribution for different methods under $1{,}000$ search queries.

We can see that RRZI_URS is almost two times more accurate than RRZI. RRZIC, RRZIC_URS, and RRZIC_MHWRS utilizing PoI count information are more accurate than RRZI and RRZI_URS. RRZIC_MHWRS samples fully accessible regions according to the PoI density distribution, which is the same as RRZIC. Meanwhile it requires less queries than RRZIC. Therefore RRZIC_MHWRS outperforms RRZIC and RRZIC_URS, which is consistent with our experimental results. Next, we evaluate the performance of our methods for estimating the cost distribution of restaurant-type PoIs. For simplicity, we divide the values of PoIs' costs into six intervals: 1) (0, 50], 2) (50, 100], 3) (100, 150], 4) (150, 200], 5) (200, 250], and 6) (250, $+\infty$). Denote by $\theta = (\theta_1, \ldots, \theta_6)$ the cost distribution, where θ_i is the fraction of PoIs with cost within the jth interval, $1 \le j \le 6$. For CDS_UNI and CDS_NOR, their associated distributions are $\theta^{(\mathrm{UNI})} = \{1/6, \ldots, 1/6\}$ and $\theta^{(\mathrm{NOR})} = (0.09, 0.016, 0.20, 0.21, 0.16, 0.18)$ respectively. Fig. 11 shows the NRMSEs of estimates of $\theta^{(\mathrm{UNI})}$ and $\theta^{(\mathrm{NOR})}$ for different methods under 1,000 queries, where we set $L = 8$. Similar to the average statistic, we can see that 1) RRZI_URS is more accurate than RRZI; 2) RRZIC_URS does not improve the accuracy of RRZIC; 3) RRZIC_MHWRS exhibits the smallest errors.

Next, we conduct experiments based on Foursquare PoI datasets. Figs. 12a, 12c, and 12e show the NRMSEs of estimates of the average statistics for different methods under 10,000 queries. We can see that RRZI_URS is almost two times more accurate than RRZI. RRZIC and RRZIC_MHWRS utilizing the PoI count information have smaller errors in comparison with RRZI and RRZI_URS. Let $\theta = (\theta_0, \theta_1, \ldots,)$ be the PoI distribution of interest. For example, θ_j, $j \ge 0$, can be defined as the fraction of PoIs with j check-ins. We define $\mathrm{RMSE}(\hat{\theta}) = \sqrt{E[\sum_{j=0}^{\infty}(\hat{\theta}_j - \theta_j)^2]}$ as a metric to measure the error of a PoI distribution estimate $\hat{\theta} = (\hat{\theta}_0, \hat{\theta}_1, \ldots,)$ with respect to its true value θ. Figs. 12b, 12d, and 12f show the RMSEs of estimates of the PoI distributions for different methods under 10,000 queries. Similarly, we can see that RRZI_URS is almost 1.5 times more accurate than RRZI. Meanwhile, RRZIC and RRZIC_MHWRS have smaller errors in comparison with RRZI and RRZI_URS. Fig. 13 shows the average number of queries required to obtain an estimate of the average number of check-ins with NRMSE less than 0.1 for our methods in comparison with the state-of-the-art method NNS [7]. We can see that our methods significantly outperform the method NNS. We omit the similar compared results for estimating the other PoI average and distribution statistics.

Table 7 shows the NRMSEs of estimates of the average number of check-ins and the RMSEs of estimates of the PoI



(a) # check-ins, average (b) # users, average (c) # tips, average

(d) # check-ins, distribution (e) # users, distribution (f) # tips, distribution

Fig. 12. (Foursquare) Compared NRMSEs and RMSEs of estimates of PoIs' average and distribution statistics for different methods under $10{,}000$ queries.

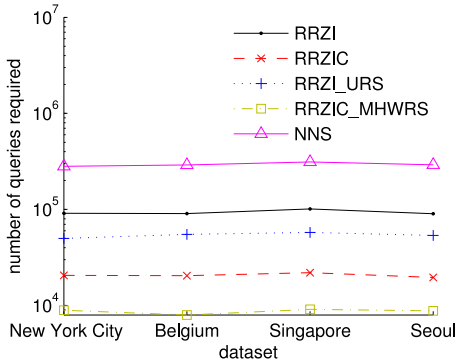


Fig. 13. (Foursquare) The number of queries required to obtain an estimate of the average number of check-ins with NRMSE less than 0.1 for our methods in comparison with NNS [7].

TABLE 8
(Our Real Application on Foursquare) Statistics of PoIs within US

Category	Fraction (%)	Average statistics (per PoI)		
		# tips	# check-ins	# users
Food	10.4	6.6	757	304
Nightlife Spot	6.4	3.4	422	166
Shop & Service	14.1	1.9	526	141
Travel & Transport	7.3	0.8	278	77
Arts & Entertainment	3.7	1.8	370	194
College & University	2.2	1.0	353	59
Outdoors & Recreation	16.0	0.7	207	64
Residence	25.8	0.2	83	5
Professional & Others	14.0	0.7	237	45

check-in count distribution for RRZI and RRZI* under 10,000 queries. We can see that RRZI* exhibits larger errors than RRZI. It may because RRZI* discards most PoIs sampled by RRZI.

5 Real Applications

In this section, we present our real applications on Foursquare, Google, and Baidu maps. The following results are obtained based on PoIs sampled on April 21-30, 2013. Foursquare provides a public API [5], which allows developers to explore a particular category of PoIs by setting the parameter "categoryId". It defines a three-level hierarchical structure of PoIs categories. The nine top-level categories are listed in Table 8. If "categoryId" is specified as a top-level category, all sub-categories will also match the query. Let $\mathbb{A}$ be the area with latitude from 26° N to 49° N and longitude from 125° W to 67° W, which covers almost the entire US territory. We apply our method RRZI_URS to sample PoIs within $\mathbb{A}$. For each top-level category, we sampled about 1.0×10^5 fully accessible regions. Table 8 shows our estimated statistics of PoIs within $\mathbb{A}$. In terms of the average statistics (i.e., the average numbers of check-ins, tips, and users), *Food*, *Nightlife Spot*, and *Shop & Service* are the top-3 popular categories. Moreover, we can see that 25.8 percent of PoIs belong to the *Residence* category, which includes four sub-categories: *Living*, *Home (Private)*, *Housing Development*, and *Residential Buildings (Appartments/Condos)*. It indicates that a large number of users have revealed their exact living places to Foursquare.

Similarly we apply our methods to sample and characterize food-type PoIs within US on Google maps. The Google public API [14] returns results including PoIs' ratings and price levels. A PoI's rating and price level are calculated based on its user reviews, where a PoI's rating ranges from 1.0 to 5.0, and a PoI's price level is an integer with the range 0 (most affordable) to 4 (most expensive). Among 4.7 million food-typed PoIs sampled, 15.5 and 18.2 percent of PoIs' ratings and price levels are available respectively. Based on these PoIs, we estimate the PoI price level distribution and find that 0.05, 72.5, 25.8, 1.6, and 0.06 percent of PoIs' price levels are 0 to 4 respectively. Fig. 14 shows our estimate of the PoI rating distribution. We can see that most PoIs have high ratings. The average rating is 3.9, and nearly 90 percent of PoIs have ratings larger than 3.0. Thus, it is hard to evaluate a PoI's relative service quality just depending on its rating value.

Next, we sample and characterize hotel-type PoIs in the following three areas on Baidu maps: 1) quadrangle region [(116° E, 40.5° N), (117° E, 39.5° N)], covering Beijing city; 2) quadrangle region [(121° E, 32° N), (122° E, 30.5° N)], covering Shanghai city; 3) quadrangle region [(113° E, 23.5° N), (114° E, 22.5 N)], covering cities Guangdong, Dongguan, and Foshan. We apply RRZIC_MHWRS to sample 1,000 fully accessible regions for each of above three regions based on the public API [15]. 51.8, 41.7, and 42.2 percent of sampled hotel-type PoIs' prices are available on Baidu maps for areas Beijing, Shanghai, and Guangzhou respectively. Based on these PoIs, we estimate the average and distribution of hotel prices. From Fig. 15, we can see that more than 20, and 50 percent of hotels' prices are in the intervals (0, 100] and (100, 200] RMB respectively, and 80 percent of hotels' prices are lower than 300 RMB. Moreover, Fig. 15 shows that Guangzhou has more hotels with price in (100, 200] RMB than Beijing and Shanghai. The average price of hotels in Beijing is 200 RMB, which is higher than Shanghai (187 RMB), and Guangzhou (168 RMB). We also use the

TABLE 7
Compared Errors of RRZI and RRZI* for Estimating the Average Value and Distribution Statistics

Dataset	Average		Distribution	
	RRZI	RRZI*	RRZI	RRZI*
New York City	0.43	1.14	0.03	0.13
Belgium	0.46	1.21	0.04	0.16
Singapore	0.50	1.31	0.03	0.12
Seoul	0.42	1.03	0.03	0.14

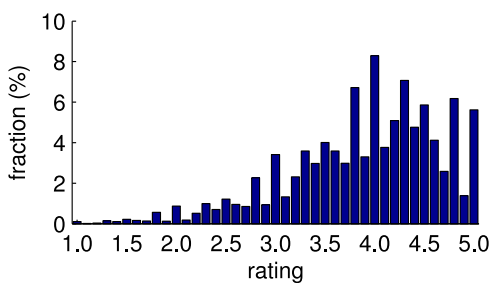


Fig. 14. (Our real application on Google maps) Rating distribution of food-type PoIs within US.

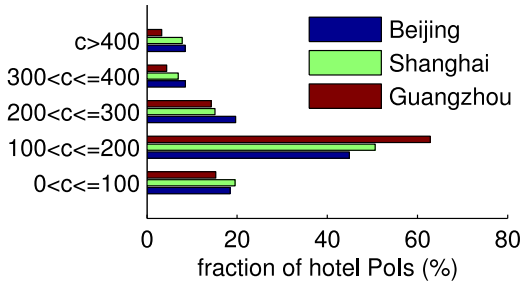


Fig. 15. (Our real application on Baidu maps) Distribution of c, hotel-type PoIs' prices per room per night.

bootstrap method to compute confidence intervals of these three estimates of average prices. The 95 percent confidence intervals of average price estimates are [169, 231] RMB, [159, 215] RMB, and [139, 197] RMB for Beijing, Shanghai, and Guangzhou respectively.

6 Related Work

Recently a lot of attention has been paid to study hidden databases using public search interfaces. Previous work focuses on crawling, retrieving, and mining information from web search engines [16], [17], [18], [19], text-based databases [20], [21], [22], [23], [24], [25], [26] and form-based [27], [28], [29], [30], [31], [32], [33], [34], [35] databases. Several sampling methods are given in [18], [36], [37], [38], [39], [40] to estimate a form-based hidden database's size (i.e., the number of tuples, refer to Khelghati et al. [41] for a good survey). These methods are designed for search engines with inputs specified as categorical data, and their performances depend on the range of input values, so they cannot be directly applied to sample PoIs using map-based search engines, which have a quite large number of input values (latitude and longitude pairs within the area of interest). To address this challenge, two methods in [7] and [8] are given to sample PoIs using public map APIs. Next we discuss them in detail.

6.1 Nearest-Neighbor Search

Dalvi et al. [7] propose a method to sample PoIs using the nearest-neighbor search API, which returns the several closest PoIs to an input location specified as a pair of latitude and longitude. Their method works as follows: At each step, it first randomly selects a point (x, y) from $\mathbb{A}$, the area of interest. Then it finds and samples the closest PoI p to (x, y) using a NNS query. Denote by $D(p)$ the points in the plane $\mathbb{A}$ that are closer to p than the other PoIs. Dalvi et al. [7] prove that their method samples PoIs with biases, and the probability of sampling p is $\gamma(p) = \frac{area(D(p))}{area(\mathbb{A})}$. To remove the sampling bias, a large number of NNS queries (e.g., on average 55 queries are used in [7]) are required to determine the boundary of $D(p)$ for each sampled PoI p. Therefore the method in [7] is quite expensive especially for service providers such as Foursquare, which does not provide a NNS API to the public.

6.2 Random Region Sampling

Li et al. [8] propose a random region sampling method to estimate the number of PoIs within $\mathbb{A}$. RRS first picks a point (x, y) from $\mathbb{A}$ at random. Then, it computes an estimate $\hat{d}$ of the PoI density around (x, y) based on the PoI densities of regions sampled previously, and initializes a new search region as a square with center (x, y) and length $\sqrt{k/\hat{d}}$, where k is the maximum number of PoIs returned for a query. If the new region has overlapping with any region sampled previously, then RRS cuts the new region to make it not collide with any sampled regions. At last RRS exhaustively searches and collects PoIs within the new region. Let $n(Q)$ be the number of PoIs within a region Q. Given $m > 0$ sampled regions Q_i $(1 \le i \le m)$, Li et al. [8] estimate $n(\mathbb{A})$ as follows $\hat{n}(\mathbb{A}) = \frac{area(\mathbb{A})}{m} \sum_{i=1}^{m} \frac{n(Q_i)}{area(Q_i)}$. We find that RRS has following drawbacks: 1) At the beginning of RRS, its PoI density estimate might exhibit a large error, so a large and dense region might be determined to explore. It requires a large number of queries to collect all PoIs within this region. 2) A very small region might be sampled from sparse areas due to RRS' region cutting operation. Then the number of PoIs within this small region is overestimated. 3) $\hat{n}(\mathbb{A})$ is not a consistent estimator of $n(\mathbb{A})$ and it might exhibit a large error. As shown in Fig. 16, $\mathbb{A}$ is specified as an $1 \times T$ quadrangle, where $T \gg 1$. $\mathbb{A}$ has two PoIs p_1 and p_2. Suppose that $k = 1$ and the distance between p_1 and p_2 is larger than 1. Let Q_1 and Q_2 be the regions sampled by RRS, which include p_1 and p_2 respectively. Clearly $area(Q_1)$ and $area(Q_2)$ are not larger than 1. Therefore, we have $\hat{n}(\mathbb{A}) \ge T \gg n(\mathbb{A}) = 2$.

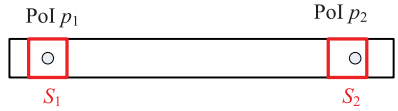


Fig. 16. An example of sampling PoIs using RRS.

7 Conclusions

In this paper, we propose methods to sample PoIs on maps, and give consistent estimators of PoI aggregate statistics. We show that the mix method RRZI_URS is more accurate than RRZI under the same number of queries used. When PoI count information is provided by public APIs, RRZIC_MHWRS utilizing this meta information is more accurate than RRZI_URS. The experimental results based on a variety of real datasets show that our methods are efficient, and they sharply reduce the number of queries required to achieve the same estimation accuracy of state-of-the-art methods.

Acknowledgments

The authors thank the anonymous reviewers as well as Prof. Bettina Kemme and Junzhou Zhao for their helpful suggestions. This work was supported in part by the National Natural Science Foundation of China (61103240, 61103241, 61221063, 91118005, 61221063, U1301254), 863 High Tech Development Plan (2012AA011003), 111 International Collaboration Program of China, the Application Foundation Research Program of SuZhou (SYG201311), and the Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute. An earlier conference

version of the paper appeared in IEEE ICDE 2014 [1]. In this journal version, we propose a new method RRZIC_MHWRS to improve the accuracy of RRZIC_URS proposed in the conference version, and conduct new experiments to evaluate the performance of our method based on a publicly available Foursquare dataset. Moreover, in this paper, we apply our methods to study characterisers of PoIs in Chinese metropolises.

REFERENCES

[1] P. Wang, W. He, and X. Liu, "An efficient sampling method for characterizing points of interests on maps," in *Proc. IEEE 30th Int. Conf. Data Eng.*, 2014, pp. 1012–1023.

[2] (2012). Google maps. [Online]. Available: https://maps.google.com/

[3] Foursquare. [Online]. Available: http://www.foursquare.com, 2012.

[4] Y. Zhu, J. Huang, Z. Zhang, Q. Zhang, T. Zhou, and Y. Ahn, "Geography and similarity of regional cuisines in china," *arXiv preprint arXiv:1307.3185*, 2013.

[5] (2013). search venues on foursqure. [Online]. Available: https://developer.foursquare.com/docs/venues/search

[6] Y. Li, M. Steiner, L. Wang, Z.-L. Zhang, and J. Bao, "Exploring venue popularity in foursquare," in *Proc. 5th IEEE Int. Workshop Netw. Sci. Commun. Netw.*, 2013, pp. 1–6.

[7] N. Dalvi, R. Kumar, A. Machanavajjhala, and V. Rastogi, "Sampling hidden objects using Nearest-neighbor oracles," in *Proc. ACM SIGKDD*, Dec. 2011, pp. 1325–1333.

[8] Y. Li, M. Steiner, L. Wang, Z.-L. Zhang, and J. Bao, "Dissecting foursquare venue popularity via random region sampling," in *Proc. ACM Conf. CoNEXT Student Workshop*, 2012, pp. 21–22.

[9] (2012). Baidu maps. [Online]. Available: http://map.baidu.com/

[10] (2013). Baidu poi datasets. [Online]. Available: http://ishare.iask.sina.com.cn/f/34170612.html

[11] S. Chib and E. Greenberg, "Understanding the Metropolis-hastings algorithm," *The Am. Statist.*, vol. 49, no. 4, pp. 327–335, Nov. 1995.

[12] W. K. Hastings, "Monte carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970.

[13] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equations of state calculations by fast computing machines," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 1087–1092, Jun. 2011.

[14] (2013). search places on google maps. [Online]. Available: https://developer.foursquare.com/docs/venues/search

[15] (2013). search places on baidu maps. [Online]. Available: http://developer.baidu.com/map/lbs-geosearch.htm

[16] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and L. C. Giles, "Methods for sampling pages uniformly from the world wide web," in *Proc. AAAI Fall Symp. Using Uncertainty Within Comput.*, Nov. 2001, pp. 121–128.

[17] Z. Bar-Yossef and M. Gurevich, "Efficient search engine measurements," in *Proc. WWW*, 2007, pp. 401–410.

[18] Z. Bar-Yossef and M. Gurevich, "Mining search engine query logs via suggestion sampling," in *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 54–65, Aug. 2008.

[19] M. Zhang, N. Zhang, and G. Das, "Mining a search engine's corpus: Efficient yet unbiased sampling and aggregate estimation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2011, pp. 793–804.

[20] J. Callan and M. Connell, "Query-based sampling of text databases," *ACM Trans. Inf. Syst.*, vol. 19, no. 2, pp. 97–130, Apr. 2001.

[21] P. G. Ipeirotis and L. Gravano, "Distributed search over the hidden web: Hierarchical database sampling and selection," in *Proc. 28th Int. Conf. Very Large Data Bases*, 2002, pp. 394–405.

[22] E. Agichtein, P. G. Ipeirotis, and L. Gravano, "Modeling query-based access to text databases," in *Proc. Int. Workshop Web Databases*, 2003, pp. 87–92.

[23] L. Barbosa and J. Freire, "Siphoning hidden-web data through keyword-based interfaces," in *Proc. SBBD*, 2004, pp. 309–321.

[24] A. Ntoulas, P. Zerfos, and J. Cho, "Downloading textual hidden web content through keyword queries," in *Proc. 5th ACM/IEEE-CS Joint Conf. Digit. Libraries*, New York, NY, USA, 2005 pp. 100–109.

[25] K. Vieira, L. Barbosa, J. Freire, and A. Silva, "Siphon++: A Hidden-webcrawler for Keyword-based interfaces," in *Proc. 17th ACM Conf. Inf. Knowl. Manage.*, 2008, pp. 1361–1362.

[26] X. Jin, N. Zhang, and G. Das, "Attribute domain discovery for hidden web databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 553–564.

[27] N. Bruno, L. Gravano, and A. Marian, "Evaluating Top-k queries over Web-accessible databases," in *Proc. Int. Conf. Data Eng.*, 2002, pp. 369–380.

[28] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro, "Crawling the content hidden behind web forms," in *Proc. Int. Conf. Comput. Sci. Its Appl.*, 2007, pp. 322–333.

[29] A. Calì and D. Martinenghi, "Querying the deep web," in *Proc. EDBT*, New York, NY, USA, 2010, pp. 724–727.

[30] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," in *Proc. 27th Int. Conf. Very Large Data Bases*, 2001, pp. 129–138.

[31] C. Sheng, N. Zhang, Y. Tao, and X. Jin, "Optimal algorithms for crawling a hidden database in the web," in *Proc. VLDB Endowment*, vol. 5, pp. 1112–1123, 2012.

[32] T. Liu, F. Wang, and G. Agrawal, "Stratified sampling for data mining on the deep web," in *Proc. IEEE 10th Int. Conf. Data Mining*, 2010, pp. 324–333.

[33] T. Liu and G. Agrawal, "Active learning based frequent itemset mining over the deep web," in *Proc. IEEE 27th Int. Conf. Data Mining*, 2011, pp. 219–230.

[34] T. Liu and G. Agrawal, "Stratified K-means clustering over a deep web data source," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1113–1121.

[35] S. Thirumuruganathan, N. Zhang, and G. Das, "Digging deeper into deep web databases by breaking through the Top-k barrier," *CoRR*, vol. abs/1208.3876, 2012.

[36] A. Dasgupta, G. Das, and H. Mannila, "A random walk approach to sampling hidden databases," in *Proc. SIGMOD*, 2007, pp. 629–640.

[37] A. Dasgupta, N. Zhang, and G. Das, "Leveraging count information in sampling hidden databases," in *Proc. IEEE Int. Conf. Data Eng.*, 2009, pp. 329–340.

[38] A. Dasgupta, X. Jin, B. Jewell, N. Zhang, and G. Das, "Unbiased estimation of size and other aggregates over hidden web databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 855–866.

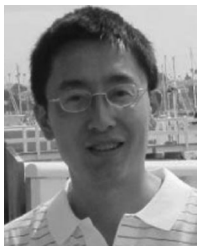
[39] A. Dasgupta, N. Zhang, and G. Das, "Turbo-charging hidden database samplers with overflowing queries and skew reduction," in *Proc. 13th Int. Conf. Extending Database Technol.*, 2010, pp. 51–62.

[40] F. Wang and G. Agrawal, "Effective and efficient sampling methods for deep web aggregation queries," in *Proc. 14th Int. Conf. Extending Database Technol.*, 2011, pp. 425–436.

[41] M. Khelghati, D. Hiemstra, and M. van Keulen, "Size estimation of Non-cooperative data collections," in *Proc. 14th Int. Conf. Inf. Integration Web-Based Appl. Serv.*, 2012, pp. 239–246.

Pinghui Wang received the BS degree in information engineering and the PhD degree in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2006 and 2012, respectively. From April 2012 to October 2012, he was a postdoctoral researcher in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. From November 2012 to October 2013, he was a postdoctoral researcher with the School of Computer Science at McGill University, QC, Canada. From January 2014 to March 2015, he was a researcher with Huawei Noah's Ark lab, Hong Kong. He is currently an associate professor in the MOE Key Laboratory for Intelligent Networks and Network Security at Xi'an Jiaotong University. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement.

Wenbo He received the PhD degree from MONET Group at the University of Illinois at Urbana-Champaign in 2008. She is an assistant professor in the School of Computer Science at the McGill University. Before that, she was an assistant professor in the EE Department at the University of Nebraska-Lincoln from 2010 to 2011, and an assistant professor in the CS Department at the University of New Mexico from 2008 to 2010. She received the Mavis Memorial Fund Scholarship Award from the College of Engineering at UIUC in 2006, and the C. W. Gear Outstanding Graduate Award in 2007 from the Department of Computer Science at UIUC. She also received the Vodafone Fellowship from 2005 to 2008, and the US National Science Foundation (NSF) TRUST Fellowship in 2007 and 2009. Recently, she received the Best Paper Awards from ACM Conference WiSec'11 and the *IEEE Transactions on Industrial Informatics*.



Xue Liu received the BS degree in mathematics and the MS degree in automatic control both from Tsinghua University, China, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2006. He is an associate professor in the School of Computer Science at McGill University, Montreal, QC, Canada. His research interests are in computer networks and communications, smart grid, real-time and embedded systems, cyber-physical systems, data centers, and software reliability. He has more than 120 publications in journals and conference proceedings in these related areas. His work has received the 2008 Best Paper Award from the *IEEE Transactions on Industrial Informatics*, and the First Place Best Paper Award from the ACM Conference on Wireless Network Security (WiSec 2011). He is a member of the ACM.