Exploiting Service Similarity for Privacy in Location-Based Search Queries

Rinku Dewri, Member, IEEE, and Ramakrishna Thurimella

Abstract—Location-based applications utilize the positioning capabilities of a mobile device to determine the current location of a user, and customize query results to include neighboring points of interests. However, location knowledge is often perceived as personal information. One of the immediate issues hindering the wide acceptance of location-based applications is the lack of appropriate methodologies that offer fine grain privacy controls to a user without vastly affecting the usability of the service. While a number of privacy-preserving models and algorithms have taken shape in the past few years, there is an almost universal need to specify one’s privacy requirement without understanding its implications on the service quality. In this paper, we propose a user-centric location-based service architecture where a user can observe the impact of location inaccuracy on the service accuracy before deciding the geo-coordinates to use in a query. We construct a local search application based on this architecture and demonstrate how meaningful information can be exchanged between the user and the service provider to allow the inference of contours depicting the change in query results across a geographic area. Results indicate the possibility of large default privacy regions (areas of no change in result set) in such applications.

Index Terms—Privacy-supportive LBS, location privacy, service quality

1 INTRODUCTION

The consumer market for location-based services (LBS) is estimated to grow from 2.9 billion dollars in 2010 to 10.4 billion dollars in 2015 [1]. While navigation applications are currently generating the most significant revenues, location-based advertising and local search will be driving the revenues going forward. The legal landscape, unfortunately, is unclear about what happens to a subscriber’s location data. The nonexistence of regulatory controls has led to a growing concern about potential privacy violations arising out of the usage of a location-based application. While new regulations to plug the loopholes are being sought, the privacy-conscious user currently feels reluctant to adopt one of the most functional business models of the decade.

Privacy and usability are two equally important requirements for successful realization of a location-based application. Privacy (location) is loosely defined as a “personally” assessed restriction on when and where someone’s position is deemed appropriate for disclosure. To begin with, this is a very dynamic concept. Usability has a two fold meaning—1) privacy controls should be intuitive yet flexible, and 2) the intended purpose of an application is reasonably maintained. Toward this end, prior research have led to the development of a number of privacy criteria, and algorithms for their optimal achievement. However, there is no known attempt to bring into view the mutual interactions between the accuracy of a location coordinate and the service quality from an application using those coordinates. Therefore, the question of what minimal location accuracy is required for an LBS application to function, remains open. The common man’s question is: “how important is my position to get me to the nearest coffee shop?”—which unfortunately remains unanswered in the scientific community.

It is worth mentioning that a separate line of research in analyzing anonymous location traces have revealed that user locations are heavily correlated, and knowing a few frequently visited locations can easily identify the user behind a certain trace [2], [3]. The privacy breach in these cases occurs because the location to identity mapping results in a violation of user anonymity. The proposal in this work attempts to prevent the reverse mapping—from user identity to user location—albeit in a user-controllable manner.

1.1 Related Work

Location obfuscation has been extensively investigated in the context of privacy. Obfuscation has been earlier achieved either through the use of dummy queries or cloaking regions. In the dummy query method, a user hides her actual query (with the true location) among a set of additional queries with incorrect locations [4], [5]. The user’s actual location is one among the locations in the query set. The additional processing overhead at the LBS, resulting from the dummy queries, must be addressed while using this method. Cheng et al. propose a data model to augment uncertainty to location data using circular regions around all objects [6]. They use imprecise queries that hide the location of the query issuer and yield probabilistic results. The results are modeled as the amount of overlap between the query range and the circular region around the queried objects. Yiu et al. propose an incremental nearest neighbor processing algorithm to retrieve...
query results [7]. The process starts with an anchor, a location different from that of the user, and it proceeds until an accurate query result can be reported. The work focuses on reducing the communication cost of the repeated querying mechanism.

Trusted third-party-based approaches rely on an anonymizer that creates spatial regions to hide the true location of users. The use of spatial and temporal cloaking to obfuscate user locations was first proposed by Gruteser and Grunwald [8]. Continuing on, Gedik and Liu develop a location privacy architecture where each user can specify maximum temporal and spatial tolerances for the cloaking regions [9]. Drawing inspiration from the concept of k-anonymity in database privacy [10], Gedik and Liu enforce a location k-anonymity requirement while creating the cloaking regions. This requirement ensures that the user will not be uniquely located inside the region in a given period of time. Chimni et al. propose a decentralized architecture to construct an anonymous spatial region, and eliminate the need for the centralized anonymizer [11]. In their approach, mobile nodes utilize a distributed protocol to self-organize into a fault-tolerant overlay network, from which a k-anonymous cloaking set of users can be determined. Kalnis et al. propose that all obfuscation methods should satisfy the reciprocity property [12]. This prevents inversion attacks where knowledge of the underlying anonymizing algorithm can be used to identify the actual object [13]. Parameter specification remains the biggest hindrance to real-world application of these techniques. Even when a user has advanced knowledge to comprehend the implications of a parameter setting on location privacy, the impact on service is unknown in these approaches. Refer to Section 1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.34, for additional literature review.

1.2 Contributions
Our contributions in this work are twofold. First, we propose a novel architecture for LBS applications that is directed toward revealing privacy/utility tradeoffs to a user before an actual geotagged query is made. Unlike a typical competitive architecture where the LBS provider does not actively participate in making privacy decisions, we envision a privacy-supportive LBS as a provider willing to provide supplemental information for making “informed” privacy decisions. An informed decision implies that the LBS user operates under reasonable knowledge about the service level implications of revealing her location with a given degree of inaccuracy. Under this platform, a user first obtains an overview of the impact of using inaccurate locations in a certain query. Thereafter, the actual query made to the service provider is geotagged with a location where the user has carefully chosen to balance result accuracy and location privacy. We describe in Section 2 the underlying rationale, setting, expectations, and components that go into such an architecture. Refer to Section 2 of the online supplementary file for a separate study, which demonstrates that users have the flexibility of adding significant noise to their locations and still obtain accurate search results.

As our second contribution, we present in Section 3, a proof of concept design for a privacy-supportive local search LBS. Given a search term (e.g., generic ones such as “cafes,” and targeted ones such as “starbucks coffee”) and a highly generalized user location (e.g., the metropolitan city), the privacy-supportive LBS generates a concise representation of the variation in the 10-nearest neighbor result set as a hypothetical user moves across the large metropolitan area. Once the representation is communicated to the user, she can infer the geographic variability that can be introduced in her location coordinates to retrieve all or a subset of the result set. Our results, using a publicly available local business database, indicate that the proposed approach can precisely reveal the area boundaries within which the result set is fully preserved (a default privacy level). Further, we observe a high degree of precision in estimating the area boundaries when user requirements on result set accuracy are relaxed (i.e., location sensitivity is hardened). Section 4 presents the empirical results to support these claims.

2 Privacy-Supportive LBS
Future LBS architectures must make room for a service provider to cooperate with the user in making sound privacy decisions. There is a growing skepticism on how a LBS provider handles (or might handle) location data. If strong market adoption is an agenda item for these businesses, then it becomes their responsibility to present evidence that the sought location accuracy is indeed a characteristic requirement of the application. Further, regulatory enforcements on location data procurement, and subsequent liability in the event of improper handling, can make the collection of unnecessarily precise geo-locations an unattractive choice. From a computational perspective, only the service provider maintains the database of queried objects in real time. Therefore, it is reasonable that differences (or similarities) in the output of a query can be efficiently computed at the server side. A user cannot make informed privacy decisions without this computation. In light of these arguments, a privacy-supportive LBS seems both appropriate and important. Note that a simple opt-in LBS is not privacy-supportive, since the implications of not using ones geolocation is not available to the user.

2.1 Setting
The communication setting we assume includes one or more users equipped with GPS-enabled devices, and an LBS provider possessing a database of points-of-interest (POI). These POI may be static, as in local business listings, or dynamic, as in a friend-finder service where users frequently check-in/out of the underlying social-networking platform. Similar to in almost all operating LBS applications, user access to the service is augmented by a geographic tag identifying the position of the user. Authentication may or may not be required to use the service, although many applications claim to be able to provide a better result set in the latter case. The service itself may require other parameters to be specified, such as search keywords or profile descriptions. The geographic tag in the query is typically the GPS-coordinates of the user device, but can also be a carefully crafted location as explained in the next section.
2.2 Architecture

The location disclosure mechanism in a privacy-supportive LBS architecture employs an intermediate communication with the LBS. A high-level schematic of the communication pattern is depicted in Fig. 1. The user device forwards the query to the LBS, albeit uses a high-level generalization of the user’s geographic location in it. This generalization may be derived as per user-specification (say at the level of the city), or obtained automatically from the location approximation that a provider can infer using a cell-towers and Wi-Fi-access points database. In response to this first query to the LBS, albeit uses a high-level generalization of the user’s geographic location in it. This generalization may be derived as per user-specification (say at the level of the city), or obtained automatically from the location approximation that a provider can infer using a cell-towers and Wi-Fi-access points database. In response to this first query phase, the user obtains a service-similarity profile. This profile is a representation of the similarities in the query output at different geographic locations. The exact form taken by this profile, as well as the data structures employed in computing this profile, may vary from application to application. A location perturbation engine on the user side then determines a noisy location to use based on the user’s privacy profile and the retrieved service-similarity profile. The LBS processes the query with respect to the noisy location. A user can manually interact with the service-similarity profile to assess which locations have the highest (or acceptable) level of result set similarity, within the constraints of the location noise she wants to infuse into the query. In this case, a good visualization of the similarity profile is required. Although this is the most flexible method of putting the tradeoff information to use, such high degree of interaction will affect the usability of the application, specially when queries are made frequently. Hence, we assume that action axioms have been provided by the user to make the process automatic. The privacy profile then states how a location is to be selected for different categories of applications, their importance, and the relative location sensitivity. Policy specifications such as these, and their integration into the decision making process, warrant an extensive exploration. We will avoid this frontier in this work. A naive approach is to allow the user to select a location sensitivity level (much like choosing the ringer-state in a mobile phone), assess query result accuracy at the corresponding location granularity (using the similarity profile), and notify the user if the accuracy drops below a threshold. Note that the policy executes within a user’s device and reveals little or no information on how locations get chosen.

2.3 Privacy Expectations and Threat Model

We interpret location privacy as the accuracy with which an adversary can determine the position of a user. This interpretation resembles the intuitive perception that a location estimated closer to our true position is more encroaching on our privacy than a relatively distant estimation. However, the privacy-supportive architecture does not make any assumption on what is “distant” and what is “close enough.” This is a significant departure from statistical measures of privacy, where a statement on “what is private” must be made proactively before issuing the query. A privacy-supportive LBS does not require this decision until the user determines the usability of the information that would be revealed as a result of the location disclosure, if at all. In light of this difference, the architecture, its underlying algorithms, or the service provider itself, cannot make any claims on the enforced level of privacy. It only facilitates the process to enforce personally desirable levels of location privacy after careful consideration of its impact. On similar grounds, we assume a threat model where the provider is semihonest (follows protocol but may be curious). Note that, on one hand, even the weakest of the adversaries may learn the precise locations of a privacy-indifferent user (one who always reveals the true location), while on the other, even the strongest of the adversaries may learn nothing additional from a privacy-paranoid user. A privacy-aware user would use the system to her advantage, perhaps frequently revealing accurate (not necessarily precise) positions, and occasionally the heavily perturbed ones. An adversary who can classify these locations as real or dummy, infers some knowledge about the user’s whereabouts—however, this is information that the user has opted to reveal in the first place.

3 A LOCAL SEARCH APPLICATION

Mobile local search is demonstrating an upward market trend, the gap with the desktop counterpart diminishing in the next three years, and then rising further. Given the penetration of web-enabled handheld devices in the consumer market, it has become exceedingly common for a user to access points database. 1 In response to this first query to the LBS, albeit uses a high-level generalization of the user’s geographic location in it. This generalization may be derived as per user-specification (say at the level of the city), or obtained automatically from the location approximation that a provider can infer using a cell-towers and Wi-Fi-access points database. 1. Creating and updating cell-towers and Wi-Fi access point maps is a costly affair. The businesses that do so (Skyhook, Google, Apple, Navizon, etc.) often consider it proprietary. The legal standard for accessing these databases is currently being litigated in a number of cases (http://epic.org/privacy/location_privacy).

to instantly look up the information she seeks to find. These search queries are estimated to produce 27.8 billion more queries than desktop-search by the year 2016. A vast majority of the users performing mobile search seek access to information pertinent in the locality of the query. Multiple LBS applications—for example, Where, AroundMe, Meet-Moi, Skout, and Loopt—have spawned in the past few years to address this market segment. In general, a local search application provides information on local businesses, events, and/or friends, weighted by the location of the query issuer. Location and service accuracy tradeoffs are clearly present in a local search LBS. A privacy-supportive variant is, therefore, well suited for this application class. Local search results tend to cycle through periods of plateaus and minor changes as one moves away from a specified location. The plateaus provide avenues for relaxation in the location accuracy without affecting service accuracy, while the minor changes allow one to assess accuracy in a continuous manner.

3.1 Problem Statement
In the traditional usage of a local search application, the user would communicate a search keyword to the provider, and retrieve a ranked list of records matching the search term. Let us denote the items that match the search term in the POI database by \( \mathcal{P} = \{P_1, P_2, \ldots, P_N\} \). A ranking function \( \mathcal{R} \) is applied to this set and a top-\( k \) subset of the ranked results is returned to the user. Since neighboring results are considered more useful, the ranking function would utilize the geolocation of the user. We use \( \mathcal{R}_k(\mathcal{P}, \text{pos}) \) to collectively denote this result set when retrieved with respect to the position \( \text{pos} \).

3.1.1 An Ideal Scenario
Let us next consider a hypothetical scenario where the user has access to a matrix that shows the percentage similarity of the result set with respect to the user’s current location. To formalize this map, let us superimpose a grid of \( r \times c \) cells on a geographic area \( \mathcal{G} \). In local search, it is sufficient to restrict focus to this geographic area while determining the set \( \mathcal{P} \). The position of the user in the grid is given as \( p = (x_0, y_0) \). Let \( \text{Sim} \) be a similarity function, defined in this application as follows:

\[
\text{Sim}(\langle x, y \rangle, \langle x', y' \rangle) = \frac{| \mathcal{R}_k(\mathcal{P}, \langle x, y \rangle) \cap \mathcal{R}_k(\mathcal{P}, \langle x', y' \rangle) |}{k}.
\]

For brevity, we will also use \( \mathcal{R}_k(\mathcal{P}, \langle x, y \rangle) \) and \( \mathcal{R}_k(\mathcal{P}, \langle x', y' \rangle) \) as arguments to the \( \text{Sim} \) function. Let \( S_{x_0,y_0} \) be a matrix of \( r \) rows and \( c \) columns, with

\[
S_{x_0,y_0}[i,j] = \text{Sim}(\langle x_0, y_0 \rangle, \langle i, j \rangle).
\]

Hence, \( S_{x_0,y_0} \) is a cell-by-cell measure of the similarity of the result set retrieved for the user’s position relative to that retrieved for any other position in the grid. As depicted in Fig. 2, this matrix allows the user to identify cell boundaries where the result set similarity gradually decreases from 100 to 0 percent. We can call them the “service-contour” of the issued query. The innermost region in the figure, \( S_{x_0,y_0} = 1.0 \), is the default privacy region—the user can claim to be anywhere in that region and yet retrieve the same result set as she would do by using her precise coordinates. The size of this default region is a characteristic feature of the distribution of the points in the set \( \mathcal{P} \) across the grid.

The service contour of a query reveals the regions where a certain percentage of the top-\( k \) results is retained. Given a certain requirement on the fraction of results that must be retained (i.e., the utility that must be maintained), the area of the corresponding region is a measure of the privacy achievable by the user, since a query originating from any point in the region will return a result set with the desired utility. The user can calculate these regions for any level of utility requirement, which in other words imply that an overall picture of the privacy/utility tradeoffs is available to the user for decision making. Trading between service accuracy and location inaccuracy is then a question of choosing a point in one of the demarcated regions.

Unfortunately, the user device cannot compute \( S_{x_0,y_0} \) without access to \( \mathcal{P} \), which resides at the LBS provider. The LBS cannot compute \( S_{x_0,y_0} \) since it requires access to the exact position \( \langle x_0, y_0 \rangle \). The question we investigate is: What form of information can the LBS provide to the user to help infer the service contour?

3.1.2 Service-Contour Inferencing
There exists a trivial solution to the raised question—push the set \( \mathcal{P} \) and the ranking function \( \mathcal{R} \) to the user, and perform the top-\( k \) ranking locally on the user device. As one can see, this solution clearly ignores underlying communication overheads and policies on sharing business intelligence. Note that the set \( \mathcal{P} \) is not simply a collection of positions, but includes additional attributes about the businesses located at those positions. This could range from names, addresses, categories, subcategories, to specifics such as value, feedback scores, and entire profiles of individuals with personal information. The ranking function \( \mathcal{R} \) is often a well-guarded business secret on how these attributes are combined. Another approach is to send a set of similarity matrices to the user, one each corresponding to a specific coordinate in the grid. The approach requires the computation and transfer of an inordinate amount of information \( (O(r^2c^2)) \). Given a geographic area, our objective is to restrict the transfer of information to a bounded
size, or \(O(1)\). The service-contour inferencing problem is then defined as follows.

**Service-contour inferencing.** Given a set of points \(P\) on a geographic area (represented as a \(r \times c\) grid), a ranking function \(R\), and a similarity function \(Sim\), find functions \(Enc\) and \(Dec\) such that

1. output \(T = Enc(P, R, Sim)\) is \(O(1)\) in size, and
2. assuming \(S'_{x,y} = Dec(T, (x,y))\), with \((x, y)\) being any point on the grid, we have \(S'_{x,y} = S_{x,y}\).

### 3.1.3 Approximate Inferencing

Without the bounded size constraint, the service-contour inferencing problem can be solved by computing the top-\(k\) results for each point in the grid, and then conveying an identification vector with respect to each point. An identification vector uniquely identifies the \(k\) results corresponding to a point. The service contour can then be exactly generated. This is an attractive choice provided the communication overhead is not exceedingly high. Note that the top-\(k\) results induce a set of order \(k\) Voronoi regions [14], [15], [16], each region sharing a certain result set. Therefore, the information to be conveyed may be highly compressible. We shall use the communication overhead of this method as a benchmark in the experimental analysis.

Consider a hypothetical scenario where the top-\(k\) results corresponding to a point can be represented by one of \(V\) symbols. Further, a maximum entropy condition is achieved under arbitrary distribution of the points in \(P\) across the grid. Therefore, each symbol is equiprobable \((1/V)\). Under this setting, no lossless compression of the symbol sequence describing the top-\(k\) results across the grid can achieve a compression level better than \(\log_2 V\) bits per point, i.e., \(rc\log_2 V\) bits for \(T\). Assuming a \(320 \times 320\) grid on a \(32 \times 32\) km\(^2\) area (a point then resembles a \(100\ m \times 100\ m\) area), and \(V = 1,000\) unique top-\(k\) result sets generated for the points in this area, this number is around 124.5 KB. While this is not a large data transfer in itself, repeated querying will result in an accumulated overhead that is a significant fraction of typical bandwidth limitations. We seek algorithms that can avoid such a communication overhead (even in the worst case); however, provide a good approximation of \(S_{x,y}\). Note that this observation assumes a worst case scenario and only pertains to the ability to correctly determine if two points have different (or the same) result sets. Computing the similarity would involve encoding additional identifier data corresponding to every set.

### 3.2 Privacy-Supported Local Search

The crucial piece of information to infer the service contour is the similarity measure \(Sim\) that tells the percentage overlap in the result sets from two points. Given that the top-\(k\) result sets (the output of \(R\)) do not always change as one moves from one point to the next, the same calculation is performed (operates on same data) by \(Sim\) for most pairs of points. Let us denote by \(V\) the set of distinct outputs of \(R\) for the points of the grid, i.e., \(V = \{R(P, (x,y)) | 1 \leq x \leq c, 1 \leq y \leq r\}\). Note that the size of \(V\) is going to be comparatively smaller than the size of the grid. Let \(V_{Sim}\) be a matrix that denotes the \(Sim\) values on pairs of elements of \(V\), i.e.,

\[
V_{Sim}[i,j] = Sim(V_i, V_j), V_i, V_j \in V.
\]

Next, we define an \(r \times c\) index matrix \(I\) such that \(I[i,j] = t\) implies \(R_{c}(P, (i,j)) = V_t\), where \(V_t\) is a member of \(V\). Fig. 3 captures the relationship between \(V, V_{Sim}\), and \(I\). In the same figure, we also see another representation of the three sets in the form of a \(5 \times 5\) pixel image. The color of each pixel is indicative of points having the same value in \(I\). In addition, the similarity measure, as computed in \(V_{Sim}\), can be inferred from the shades of the colors

\[
Sim((x, y), (x', y')) = 1 - |\text{color}(x, y) - \text{color}(x', y')|.
\]

For example, the result set similarity between the points \((3, 3)\) and \((5, 5)\) is \(V_{Sim}[2, 3] = 0.4\), which can also be derived as \(1 - [0.6 - 0.0]\). The advantage here is that the similarity information is conveyed without the need to communicate \(V\). The representation is rather straightforward in this example, but need not be so for arbitrary \(V, V_{Sim}\), and \(I\).

#### 3.2.1 Multidimensional Scaling

The example above involves determining three grayscale color codes (values in \([0, 1]\)) such that the euclidean distance between two values is proportional to the similarity measurements given by \(V_{Sim}\). The objective is not different when \(V_{Sim}\) has a significantly more number of entries. We adopt the classical method of multidimensional scaling at this step. The multidimensional scaling problem is stated as follows for the problem at hand.

**Multidimensional scaling.** Given a set of top-\(k\) result sets \(V = \{V_1, V_2, \ldots, V_n\}\) and a similarity matrix \(V_{Sim}\), obtain a set of \(n\) \(m\)-dimensional vectors \(c_1, c_2, \ldots, c_n\) that minimizes

\[
\sum_{i < j} (\text{Euc}(c_i, c_j) - (1 - V_{Sim}[i,j])^2).
\]

\(\text{Euc}\) is the euclidean distance function. The scaling happens from a \(k\)-dimensional space to an \(m\)-dimensional space. For the case when a minimum value of zero exists (and is found), the euclidean distance between any two vectors \(c_i\) and \(c_j\) is equal to the dissimilarity between two result sets \(V_i\) and \(V_j\). Such distance preserving embedding of high-dimensional data is readily useful for data visualization.
Numerical solvers for a multidimensional scaling problem are included in most statistical packages. We use the implementation provided in the cmdscale function of the R statistical package. The implementation follows the analysis of Mardia [17]. We use a value of $m = 3$ since it allows one to graphically visualize the similarity trend in the form of an RGB color image. Higher values of $m$ allow for the possibility of better distance preservation, but results in a larger encoded size.

The $Enc$ function based on 3D scaling then operates as follows: each component of the $c_i$ vectors are normalized to the $[0, 1]$ interval, and an $r \times c$ pixel image is created with the RGB color of pixel $(i, j)$ set to $c_{[i,j]}$. This image is the output $T$ produced by the $Enc$ function and communicated to the user. Although a vector $c_i$ can take infinite values in $[0, 1]^3$, the number of possibilities reduces to 16.7 million due to the color mapping. Fig. 1 in Appendix A (see the online supplementary file) illustrates an example image created by $Enc$ for 10-nearest Starbucks coffee shop locations in the city of Los Angeles, CA (1,024 square kilometers area centered around Los Angeles City Hall).

### 3.2.2 Inferring the Service Contour

To retrieve the service contour from $T$, the $Dec$ function uses the location of the user $(x_0, y_0)$ as a point of reference for similarity comparison. Let $T_{x,y}$ be the RGB color vector at the $(x, y)$ pixel in $T$. The euclidean distance between $T_{x_0,y_0}$ and the color vector $T_{i,j}$ of any other pixel $(i, j)$ (a point in the grid) attempts to closely estimate the dissimilarity measure—the similarity estimate then being $S_{x_0,y_0}[i,j] = 1 - \text{Euc}(T_{x_0,y_0}, T_{i,j})$. The $Dec$ function then simply computes this estimate for all possible points $(i, j)$ in the grid. Computation of the service-contour can also be parameterized by a threshold $\delta$ such that points in the grid with a similarity estimate higher or equal to $\delta$ are the only ones identified. To do so, one can begin at point $(x_0, y_0)$ and continue to explore neighboring points as long as the similarity estimate satisfies the threshold. We explore three fast heuristics to avoid a point by point generation of the service contour. Fig. 4 illustrates the difference between them.

**Box.** Starting from the user location $(x_0, y_0)$, a box is grown by pushing the four edges outward (in clockwise order), one point-step at a time. Edge pushing along a direction is stopped whenever doing so will result in the inclusion of a point with similarity estimate less than $\delta$.

**Inscribed circle.** Box expansion tends to cover inaccurate points (those outside the threshold) in the corner areas, specially when similarity estimates are not exact. A circular region inscribed in the box, centered at $(x_0, y_0)$, eliminates such errors on the corners of the box.

**Fill-out.** While an inscribed circle is good at reducing the error in some cases, it cannot cover irregular shaped regions within the $\delta$ threshold. The fill-out method expands the circular region by including neighboring points that has the same color vectors as points within the inscribed circle.

An interactive process of inference would involve determining the service contour for a given value of $\delta$ (say 90 percent), and then progressively growing it depending on the area of the region inferred at a certain threshold.

We refrain from using methods based on computational geometry due to their higher processing requirements.

Note that we have excluded the possibility of a malicious server model in this scheme. A malicious server can manipulate the similarity data to create the impression that no two neighboring cells have the same result set. However, it would not be correct to state that such manipulations will force the user to reveal her precise location. The decision on whether a default privacy region is sufficiently large enough is user driven. A distorted picture of the similarity profile may in fact drive the user to believe that no reasonable privacy can be achieved in the application, and thereby discontinue using it. In another case, a privacy-aware user may still pick a location from a larger area, i.e., trade accuracy (although based on distorted information) for privacy. Hence, even after a malicious server manipulates the similarity matrix intelligently, it is not guaranteed that the location communicated by the user is true, or a consequence of the privacy/accuracy tradeoff process. In addition, the server must also keep the user motivated to use the service. This in itself is much more difficult once the user observes discrepancies in the final query answers and the physical realities. A formal evaluation substantiating these arguments would be useful; otherwise, distributed methods to share trust scores on service providers can be sought to identify malicious servers.

### 4 Empirical Evaluation

The empirical evaluation is performed using the SimpleGeo Places data set that contains information on more that 20 million places around the world, and distributed under the Creative Commons open license. The US part of the data set has 12,993,248 entries, with data corresponding to multiple business categories and subcategories. Entries are maintained in the GeoJSON format, and includes attributes such as name, latitude/longitude, address, phone numbers, classifiers (category, type, subcategory) and tags. In our study, a place is considered a match for the search keyword.
if it includes the keyword in any of these attributes, and the
city matches the city attribute. The evaluation is performed
for the four largest cities in USA—Los Angeles, Houston,
Chicago, and New York. One of the factors influencing the
top-\(k\) results is the number of objects returned by a query,
and their distribution around the query point. The existence
of a large number of objects implies that the top-\(k\) results
are likely to change for small changes in location. For
objects that are low in density, large variations in the
location are possible without changing the result set. This
behavior can be reasonably assumed irrespective of the
density of users in the city. Therefore, we choose large cities
where we can obtain different densities of objects, specially
ones with high densities. Objects that are high in density in
large cities may not be so in a smaller city. Hence, we
believe that a comprehensive evaluation can be performed
by considering these large cities.

For each city, a 1,024 km\(^2\) area is used as the high-level
generalization \(g\) to generate the similarity profile. A 320 \(\times\)
320 cells grid is superimposed on this area. Each cell then
reflects a generalization. This approach implicitly assumes
that positioning a user in a cell is equivalent to exactly locating her. For Los Angeles and Houston, the city
center is at the center of this grid \((160, 160)\). For Chicago
and New York, the city centers are at \((288, 160)\) and
\((32, 160)\), respectively. The geographic coordinates are
provided in Appendix A, which is available in the online
supplemental material. Euclidean distance-based nearest
neighbor is used as the ranking function, with \(k = 10\). We
employ the cover tree algorithm by Beygelzimer et al. [18]
to determine the 10 nearest query matches with respect to a
point on the grid.

Instead of experimenting with a large corpus of search
keywords, we generalize the notion of query points into
low-, medium-, and high-density objects. Low-density
objects result from targeted queries, with frequencies
ranging from 10 to 50 within the grid. Queries resulting in
50 to 200 objects are considered medium density, while
frequencies higher than that are considered high density.
We were able to generate low-density objects by using
search terms such as “bowling,” “electronics store” and
local grocery store names in the cities. Medium density
objects are generated from search terms such as “starbucks
coffee” and “police.” High-density objects are generated by
heavily generic terms such as “atm” and “gas station.” For
the high-density case, frequencies were often observed to be
in the range of 400 to 900. The search keyword itself does
not hold much importance for this study, but is used to
retrieve query point distributions that reflect the real world.
The results below combine performance measures irrespective
of what search term produced them, the only distinction being made is with respect to the density.

### 4.1 Evaluation Process

Performance of the Enc and Dec functions is measured
using precision and recall metrics. Given a threshold \(\delta\), we
arrive at a set of points \(Z\) on the grid that the user can use to
perturb her location. Depending on the accuracy of
maintaining similarities, and the subsequent estimation by
the three heuristics, this set of points may be over or
underestimated. If \(Z_{\text{true}}\) is the true set of points satisfying
the threshold, then the precision is given as the fraction of
points in \(Z\) that are also in \(Z_{\text{true}}\). Recall is the percentage of
points in \(Z_{\text{true}}\) that are also in \(Z\).

\[
\text{Precision} = \frac{|Z \cap Z_{\text{true}}|}{|Z|}; \quad \text{Recall} = \frac{|Z \cap Z_{\text{true}}|}{|Z_{\text{true}}|}.
\]

Precision can be viewed as the probability that the
service similarity guarantee (within the threshold) is not
violated. Recall measures the ability to identify the areas
where a certain level of service similarity is guaranteed.
While precision can be viewed as a measure of the quality
of service, the absolute recalled area \(|Z \cap Z_{\text{true}}|\) is the size of
the geographic region where the user can hide herself,
yet retrieve true query results (within the threshold). In
other words, the recall-area may be viewed as a measure of
the privacy level obtained by the user.

Experiments are performed for four service similarity
thresholds: \(\delta = 1.0, 0.9, 0.8, \) and \(0.7\). For each value, precision
and recall are calculated for the three heuristics using a
sample of points as the user location \((x_0, y_0)\) on the grid.
The sample consists of 1,521 points uniformly distributed
on the grid—a sample point every 800 m (0.5 mi) along the
horizontal and vertical directions. For \(\delta = 1.0\), results are
only reported for the fill-out heuristic.

#### 4.2 The Case of “Starbucks Coffee”

The case of locally searching a coffee shop—for example,
“starbucks coffee”—often comes up in location privacy
discussions. We present the detailed comparative results
with respect to a privacy-aware user trying to find the
nearest Starbucks coffee shop location. Figs. 5 and 6 show
the comparative efficiency of the three heuristics in the four
cities. For each city, the precision and recall plots show the
performance of fill-out for \(\delta = 1.0\) (leftmost) and then three
sets of rectangles, one each for \(\delta = 0.9, 0.8, \) and \(0.7\) (from left
to right). A precision and recall of 1.0 for fill-out at \(\delta = 1.0\)
implies that a privacy-indifferent user does not lose any
accuracy in the result set as a result of the process. In
addition, the heuristic exactly reveals the default privacy
region with respect to the issued query. For the other \(\delta\)
values, each rectangle shows the 10th percentile (lower
eedge), 25th percentile (center dot), and 50th percentile
(upper edge) of the computed precision and recall values.
Recall that the \(p\)th percentile is the value below which \(p\)
percentage of the observations lie. The inscribed-circle and
fill-out heuristics guarantee 90 percent or more precision for
75 percent (25th percentile) of the points sampled on the
grid (possible user locations), across the four cities. This is
observed irrespective of the service similarity requirement
imposed by a user. Precision for the box heuristic is
comparatively worse because of its tendency toward
erroneous inclusion of points. As expected, inscribed circle
clearly improves upon this, but results in an extensive
pruning of the identified points (poor recall). It is not
difficult to create a heuristic with high precision; however,
the desirable one has high recall as well.

Fill-out improves upon the recall of inscribed circle
without heavily degrading the precision. However, the
call values themselves are all below 50 percent. The
bottom of each plot shows trend lines depicting how the
area recalled ($|Z \cap Z_{\text{true}}|$ in km$^2$) by the fill-out heuristic changes as a user moves away from the city center. The query object (“starbucks coffee”) has a relatively higher concentration near the city center areas. The trend line for $\delta = 1.0$ (for which fill-out has 100 percent recall) indicates that the default privacy region may not be significantly large when query objects are concentrated. However, areas as large as 20-40 km$^2$ become available within 8 km ($\approx$ 5 mi) of the city center, provided one or two incorrect results are acceptable. This is despite the poor recall of the heuristic. These areas will presumably be large enough for a privacy-conscious user, given that the observations hold more strongly for regions that see lesser crowd. Note that changing the service accuracy requirement further down can expand the determined area. Object locations in this case, although not the nearest ones, will not be unrealistically far away.

### 4.3 Precision/Recall Trends

The precision and recall trends we observe for the case of “starbucks coffee” are repeated for the other medium density experiment (derived using the keyword “police”). For the fill-out heuristic, Fig. 7 shows the mean (across the search keywords) of the 25th percentiles of the precision scores for different object densities. Full precision for low-density objects is almost guaranteed, irrespective of the service accuracy threshold. However, the approach has difficulty maintaining those same values for high-density objects. High-density objects are often located close to each other, thereby creating a scenario where moving small...
distances significantly changes the result set. It also means that finding such objects is not difficult in the real world. Note that the density designation is not based on what is being queried—a “gas station” could be a high-density object in parts of a city, and low/medium in others. In the latter case, when finding one could become difficult by simply looking around, local search is possible in a privacy-supportive manner. The ranking function is also a crucial component in deciding the density of objects. For instance, a ranking function that accounts for local reviews of restaurants while making suggestions, will result in a low density categorization for the keyword “restaurants,” meaning the top-$k$ result set does not change significantly even for a high concentration of restaurants in the area.

The recalled area is also significantly large for low-density objects, occasionally dropping when clusters of such objects are found. Fig. 8 depicts this drop for the cities of Chicago and New York. The observation reinstates the fact that object densities can be locally high. The conclusions made in the “starbucks coffee” case remains applicable in general to the recalled area for medium density objects. Refer to Section 3 in the online supplementary file for results on the communication overhead associated with the proposed methodology.

4.4 Conclusions

Based on the observations from the empirical study, we make the following conclusions on the efficacy of a privacy-supportive local search application.

Precise geolocations are necessary for result set accuracy when the queried objects exist as a dense cluster in the search area. It seems unlikely that both location privacy and result exactness can be maintained in this case. A privacy-supportive application would allow the user to aggressively tradeoff the service similarity requirement to determine a sufficiently large area for location perturbation. Given the high density of objects, resulting objects can still be expected to be in the near vicinity.

When object density is not dense, location accuracy has a minor role to play in retrieving relevant results. A privacy-supportive application would help identify the large default-privacy regions resulting in such situations.

Next generation telecommunication systems could very well make it possible to quickly (and cost-effectively) transfer all information required to infer the service contour exactly. Until then, approximate inferencing algorithms can be used to reduce the communication overhead.

5 Summary

In this paper, we proposed a novel architecture to help identify privacy and utility tradeoffs in an LBS. The architecture has a user-centric design that delays the sharing of a location coordinate until the user has evaluated the impact of its accuracy on the service quality. Using the prototypical example of a local search application, we showed the form of information that can be exchanged between the user and the provider to enable a privacy-supportive LBS. Section 4 of the online supplementary file suggests some future directions of research for this work.

REFERENCES


Rinku Dewri received the PhD degree in computer science from Colorado State University. He is an assistant professor in the Computer Science Department at the University of Denver. His research interests include the area of information security and privacy, risk management, data management, and multicriteria decision making. He is a member of the IEEE and ACM.

Ramakrishna Thurimella received the PhD degree in computer science from the University of Texas, Austin, and has more than 20 years of experience in the academia. He is a professor and chair of the Computer Science Department at the University of Denver. His research interests include algorithms, and information security and privacy.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.