Exploration of Interesting Dense Regions on Spatial Data

Plácido A. Souza Neto
Federal Institute of Rio Grande do Norte (Brazil)
placido.neto@ifrn.edu.br

Francisco B. Silva Júnior
Federal Institute of Rio Grande do Norte (Brazil)
bento.francisco@academico.ifrn.edu.br

Felipe F. Pontes
Federal Institute of Rio Grande do Norte (Brazil)
freire.pontes@academico.ifrn.edu.br

Behrooz Omidvar-Tehrani
University of Grenoble Alpes (France)
behrooz.omidvar-tehrani@univ-grenoble-alpes.fr

ABSTRACT

Nowadays, spatial data are ubiquitous in various fields of science, such as transportation and the social Web. A recent research direction in analyzing spatial data is to provide means for "exploratory analysis" of such data where analysts are guided towards interesting options in consecutive analysis iterations. Typically, the guidance component learns analyst's preferences using her explicit feedback, e.g., picking a spatial point or selecting a region of interest. However, it is often the case that analysts forget or don't feel necessary to explicitly express their feedback in what they find interesting. Our approach captures implicit feedback on spatial data. The approach consists of observing mouse moves (as a means of analyst's interaction) and also the explicit analyst's interaction with data points in order to discover interesting spatial regions with dense mouse hovers. In this paper, we define, formalize and explore Interesting Dense Regions (IDRs) which capture preferences of analysts, in order to automatically find interesting spatial highlights. Our approach involves a polygon-based abstraction layer for capturing preferences. Using these IDRs, we highlight points to guide analysts in the analysis process. We discuss the efficiency and effectiveness of our approach through realistic examples and experiments on Airbnb and Yelp datasets.

1 INTRODUCTION

Nowadays, there has been a meteoric rise in the generation of spatial datasets in various fields of science, such as transportation, lodging services, and social science. As each record in spatial data represents an activity in a precise geographical location, analyzing such data enables discoveries grounded on facts. Analysts are often interested to observe spatial patterns and trends to improve their decision making process. Spatial data analysis has various applications such as smart city management, disaster management, and autonomous transport [28, 30].

Typically, spatial data analysis begins with an imprecise question in the mind of the analyst, i.e., *exploratory analysis*. The analyst requires to go through several trial-and-error iterations to improve her understanding of the spatial data and gain insights. Each iteration involves visualizing a subset of data on geographical maps using an off-the-shelf product (e.g., Tableau[1], Exhibit[2], Spotfire[3]) where the analyst can investigate on different parts of the visualization by zooming in/out and panning.

[1] *http://www.tableau.com*
[2] *http://www.simile-widgets.org/exhibit/*
[3] *http://spotfire.tibco.com*

Spatial data are often voluminous. Hence the focus in the literature of spatial data analysis is on "efficiency", i.e., enabling fluid means of navigation in spatial data to facilitate the exploratory analysis. The common approach is to design pre-computed indexes which enable efficient retrieval of spatial data (e.g., [23, 34]). However, there has been less attention to the "value" derived from spatial data. Despite the huge progress on the efficiency front, an analyst may easily get lost in the plethora of geographical points due to two following reasons.

■ In an exploratory context, the analyst doesn't know a priori what to investigate next.

■ Moreover, she may easily get distracted and miss interesting points by visual clutter caused by huge point overlaps.

The main drawback of the traditional analysis model is that the analyst has a *passive role* in the process. In other words, the analyst's feedback (i.e., her likes and dislikes) is ignored and only the input query (i.e., her explicit request) is served. In case feedback is incorporated, the process can be more directed towards analyst's interests where her partial needs can be served earlier in the process. In this paper, we advocate for a "guidance layer" on top of the raw visualization of spatial data to enable analysts know *"what to see next"*. This guidance should be a function of analyst feedback: the system should return options similar to what the analyst has already appreciated.

Various approaches in the literature propose methodologies to incorporate analyst's feedback in the exploration process of spatial data [4, 8, 24, 33]. Typically, feedback is considered as a function which is triggered by any analyst's action on the map. The action can be "selecting a point", "moving to a region", "asking for more details", etc. The function then updates a "profile vector" which keeps tracks of analyst's interests. The updated content in the profile vector enables the guidance functionality. For instance, if the analyst shows interest in a point which describes a house with balcony, this choice of amenity will reflect her profile to prioritize other houses with balcony in future iterations.

Feedback is often expressed *explicitly*, i.e., the analyst clicks on a point and mentions if she likes or dislikes the point [19, 25, 26]. In [26], we proposed an interactive approach to exploit such feedback for enabling a more insightful exploration of spatial data. However, there are several cases that the feedback is expressed *implicitly*, i.e., the analyst does not explicitly click on a point, but there exist correlations with other signals captured from the analyst which provide hint on her interest. For instance, it is often the case in spatial data analysis that analysts look at some regions of interest but do not provide an explicit feedback. Another example is frequent mouse moves around a region which is a good indicator of the analyst's potential interest in the points in that region. Implicit feedbacks are more challenging to capture and hence less investigated in the literature. The following

example describes a use case of implicit feedbacks. This will be our running example which we follow throughout the paper.

Example. *Benício is planning to live in Paris for a season. He decides to rent a home-stay from Airbnb website*[4]*. He likes to discover the city, hence he is open to any type of lodging in any region with an interest to stay in the center of Paris. The website returns 1500 different locations. As he has no other preferences, an exhaustive investigation needs scanning each location independently which is nearly infeasible. While he is scanning few first options, he shows interest in the region of Trocadero (where the Eiffel tower is located at) but he forgets or doesn't feel necessary to click a point there. An ideal system should capture this implicit feedback in order to short-list a small subset of locations that Benício should consider as high priority.*

The above example shows in practice that implicit feedback capturing is crucial in the context of spatial data analysis. While text-boxes, combo-boxes and other input elements are available in analyzing other types of data, the only interaction means between the analyst and a spatial data analysis system is a geographical map spanned on the whole screen. In this context, a point can be easily remained out of sight and missed.

In this paper, we present an approach whose aim is to capture and analyze implicit feedback of analysts in spatial data analysis. Without loss of generality, we focus on "mouse moves" as the implicit feedback received from the analyst. Mouse moves are the most common way that analysts interact with geographical maps [11]. It is shown in [3] that mouse gestures have a strong correlation with "user engagement". Intuitively, a point gets a higher weight in the analyst's profile if the mouse cursor moves around it frequently. However, our approach can be easily extended to other types of inputs such as gaze tracking, leap motions, etc.

Contributions. In this paper, we make the following contributions:

- We define and explore the notion of "implicit user feedback" which enables a seamless navigation in spatial data;
- We define the notion of "information highlighting", a mechanism to highlight out-of-sight important information for analysts. A clear distinction of our proposal with the literature is that it doesn't aim for pruning (such as top-k recommendation), but leveraging the actual data with potential interesting results (i.e., highlights);
- We define and formalize the concept of Interesting Dense Regions (IDRs), a polygon-based approach to explore and highlight spatial data;
- We propose an efficient greedy approach to compute highlights on-the-fly;
- We show the effectiveness of our approach through a set of qualitative experiments.

The outline of the paper is the following. Section 2 describes our data model. In Section 3, we formally define our problem. Then in Section 4, we present our solution and its algorithmic details. Section 5 reports our experiments on the framework. We review the related work in Section 6. We present some limitations of our work in Section 7. Last, we conclude in Section 8.

2 DATA MODEL

We consider two different layers on a geographical map: "spatial layer" and "interaction layer". The spatial layer contains points from a spatial database $\mathcal{P}$. The interaction layer contains mouse move points $\mathcal{M}$.

Spatial layer. Each point $p \in \mathcal{P}$ is described using its coordinates, *latitude* and *longitude*, i.e., $p = \langle lat, lon \rangle$. Note that in this work, we don't consider "time" for spatial points, as our contribution focuses on their location. Points are also associated to a set of domain-specific attributes $\mathcal{A}$. For instance, for a dataset of a real estate agency, points are properties (houses and apartments) and $\mathcal{A}$ contains attributes such as "surface", "number of pieces" and "price". The set of all possible values for an attribute $a \in \mathcal{A}$ is denoted as $dom(a)$. We also define analyst's feedback F as a vector over all attribute values (i.e., facets), i.e., $F = \overrightarrow{\cup_{a \in \mathcal{A}} dom(a)}$. The vector F is initialized by zeros and will be manipulated to express analyst's preferences.

Interaction layer. Whenever the analyst moves her mouse, a new point m is appended to the set $\mathcal{M}$. Each mouse move point is described using the pixel position that it touches and the clock time of the move. Hence each mouse move point is a tuple $m = \langle x, y, t \rangle$, where x and y specifies the pixel location and t is a Unix Epoch time. To conform with geographical standards, we assume $m = \langle 0, 0 \rangle$ sits at the middle of the interaction layer, both horizontally and vertically.

The analyst is in contact with the interaction layer. To update the feedback vector F, we need to translate pixel locations in the interaction layer to latitudes and longitudes in the spatial layer. While there is no precise transformation from planar to spherical coordinates, we employ equirectangular projection to obtain the best possible approximation. Equation 1 describes this formula to transform a point $m = \langle x, y, t \rangle$ in the interaction layer to a point $p = \langle lat, lon \rangle$ in the spatial layer. Note that the resulting p is not necessarily a member of $\mathcal{P}$.

$$lon = \frac{x}{cos\gamma} + \theta; lat = y + \gamma \tag{1}$$

The inverse operation, i.e., transforming from the spatial layer to the interaction is done using Equation 2.

$$x = (lon - \theta) \times cos\gamma; y = lat - \gamma \tag{2}$$

The reference point for the transformation is the center of both layers. In Equations 1 and 2, we assume that γ is the latitude and θ is the longitude of a point in the spatial layer corresponding to the center of the interaction layer, i.e., $m = \langle 0, 0 \rangle$.

3 PROBLEM DEFINITION

The large size of spatial data hinders its effective analysis for discovering insights. Analysts require to obtain only few options (so-called "highlights") to focus on. These options should be in-line with what they have already appreciated. In this paper, we formulate the problem of "information highlighting using implicit feedback", i.e., highlight few spatial points based on implicit interests of the analyst in order to guide her towards what she should concentrate on in consecutive iterations of the analysis process. We formally define our problem as follows.

Problem. *Given a time t_c and an integer constant k, obtain an updated feedback vector F using points $m \in \mathcal{M}$ where $m.t \leq t_c$ and choose k points $\mathcal{P}_k \subseteq \mathcal{P}$ as "highlights" where $\mathcal{P}_k$ satisfies two following constraints.*

- $\forall p \in \mathcal{P}_k$, *similarity*$(p, F)$ *is maximized.*
- *diversity*$(\mathcal{P}_k)$ *is maximized.*

[4] *http://www.airbnb.com*

Algorithm 1: Spatial Highlighting Algorithm

Input: Current time t_c, mouse move points $\mathcal{M}$
Output: Highlights $\mathcal{P}_k$

1 $\mathcal{S} \leftarrow$ *find_interesting_dense_regions*$(t_c, \mathcal{M})$
2 $\mathcal{P}_s \leftarrow$ *match_points*$(\mathcal{S}, \mathcal{P})$
3 $F \leftarrow$ *update_feedback_vector*$(F, \mathcal{P}_s)$
4 $\mathcal{P}_k \leftarrow$ *get_highlights*$(\mathcal{P}, F)$
5 **return** $\mathcal{P}_k$

The first constraint guarantees that returned highlights are highly similar with analyst's interests captured in F. The second constraint ensures that k points cover different regions and they don't repeat themselves. While our approach is independent from the way that *similarity* and *diversity* functions are formulated, we provide a formal definition of these functions in Section 4.

The aforementioned problem is hard to solve due to the following challenges.

■ **Challenge 1.** First, it is not clear how mouse move points influence the feedback vector. Mouse moves occur on a separate layer and there should be some meaningful transformations to interpret mouse moves as potential changes in the feedback vector.

■ **Challenge 2.** Even if an oracle provides a mapping between mouse moves and the feedback vector, analyzing all generated mouse moves is challenging and may introduce false positives. A typical mouse with 1600 DPI (Dots Per Inch), touches 630 pixels for one centimeter of move. Hence a mouse move from the bottom to the top of a typical 13-inch screen would provide 14,427 points which may not be necessarily meaningful.

■ **Challenge 3.** Beyond two first challenges, finding the most similar and diverse points with F needs an exhaustive scan of all points in $\mathcal{P}$ which is prohibitively expensive: in most spatial datasets, there exist millions of points. Moreover, we need to follow multi-objective considerations as we aim to optimize both similarity and diversity at the same time.

We recognize the complexity of our problem using the aforementioned challenges. In Section 4, we discuss a solution for the discussed problem and its associated challenges.

4 INTERESTING DENSE REGIONS

Our approach exploits analyst's implicit feedback (i.e., mouse moves) to highlight few interesting points as future analysis directions. Algorithm 1 summarizes the principled steps of our approach.

The algorithm begins by mining the set of mouse move points $\mathcal{M}$ in the interaction layer to discover one or several Interesting Dense Regions, abbr., IDRs, in which most analyst's interactions occur (line 1). Then it matches the spatial points $\mathcal{P}$ with IDRs using Equation 2 in order to find points inside each region (line 2). The attributes of resulting points will be exploited to update the analyst's feedback vector F (line 3). The updated vector F will then be used to find k highlights (line 4). These steps ensure that the final highlights reflect analyst's implicit interests. We detail each step as follows.

Algorithm 2: Find Interesting Dense Regions (IDRs)

Input: Current time t_c, mouse move points $\mathcal{M}$
Output: IDRs $\mathcal{S}$

1 $\mathcal{S} \leftarrow \emptyset$
2 $g \leftarrow$*number of time segments*
3 **for** $i \in [0, g]$ **do**
4 $\quad \mathcal{M}_i \leftarrow \{m = \langle x, y, t\rangle | (\frac{t_c}{g} \times i) \leq t \leq (\frac{t_c}{g} \times (i+1))\}$
5 $\quad C_i \leftarrow$ *mine_clusters*$(\mathcal{M}_i)$
6 $\quad O_i \leftarrow$ *find_ploygons*(C_i)
7 **end**
8 **for** O_i, O_j *where* $i, j \in [0, g]$ *and* $i \neq j$ **do**
$\quad \mathcal{S}.append(intersect(O_i, O_j))$
9 **return** $\mathcal{S}$

4.1 Discovering IDRs

The objective of this step is to obtain one or several regions in which the analyst has expressed her implicit feedback. There are two observations for such regions.

■ **Observation 1.** We believe that a region appeals more interesting to the analyst if it is denser, i.e., the analyst moves her mouse in that region several times.

■ **Observation 2.** It is possible that the analyst moves her mouse everywhere in the map. This should not signify that everywhere in the map has the same significance.

Following our observations, we propose Algorithm 2 for mining IDRs. We add points to $\mathcal{M}$ only every 200ms to prevent adding redundant points (i.e., Challenge 2). Following Observation 1 and in order to mine the recurring behavior of the analyst, the algorithm begins by partitioning the set $\mathcal{M}$ into g fixed-length consecutive segments $\mathcal{M}_0$ to $\mathcal{M}_g$. The first segment starts at time zero (where the system started), and the last segment ends at t_c, i.e., the current time. Following Observation 2, we then find dense clusters in each segment of $\mathcal{M}$ using a variant of DB-SCAN approach [15]. Finally, we return intersections among those clusters as IDRs.

For clustering points in each time segment (i.e., line 5 of Algorithm 2), we use ST-DBSCAN [9], a space-aware variant of DB-SCAN for clustering points based on density. For each subset of mouse move points $\mathcal{M}_i$, $i \in [0, g]$, ST-DBSCAN begins with a random point $m_0 \in \mathcal{M}_i$ and collects all density-reachable points from m_0 using a distance metric. As mouse move points are in the 2-dimensional pixel space (i.e., the display), we choose euclidean distance as the distance metric. If m_0 turns out to be a core object, a cluster will be generated. Otherwise, if m_0 is a border object, no point is density-reachable from m_0 and the algorithm picks another random point in $\mathcal{M}_i$. The process is repeated until all of the points have been processed.

Once clusters are obtained for all subsets of $\mathcal{M}$, we find their intersections to locate recurring regions (line 6). To obtain intersections, we need to clearly define the spatial boundaries of each cluster. Hence for each cluster, we discover its corresponding polygon that covers the points inside. For this aim, we employ Quickhull algorithm, a quicksort-style method which computes the convex hull for a given set of points in a 2D plane [6].

We describe the process of finding IDRs in an example. Figure 1 shows the steps that Benício follows in our running example to explore home-stays in Paris. Figure 1.A shows mouse movements of Benício in different time stages. In this example, we

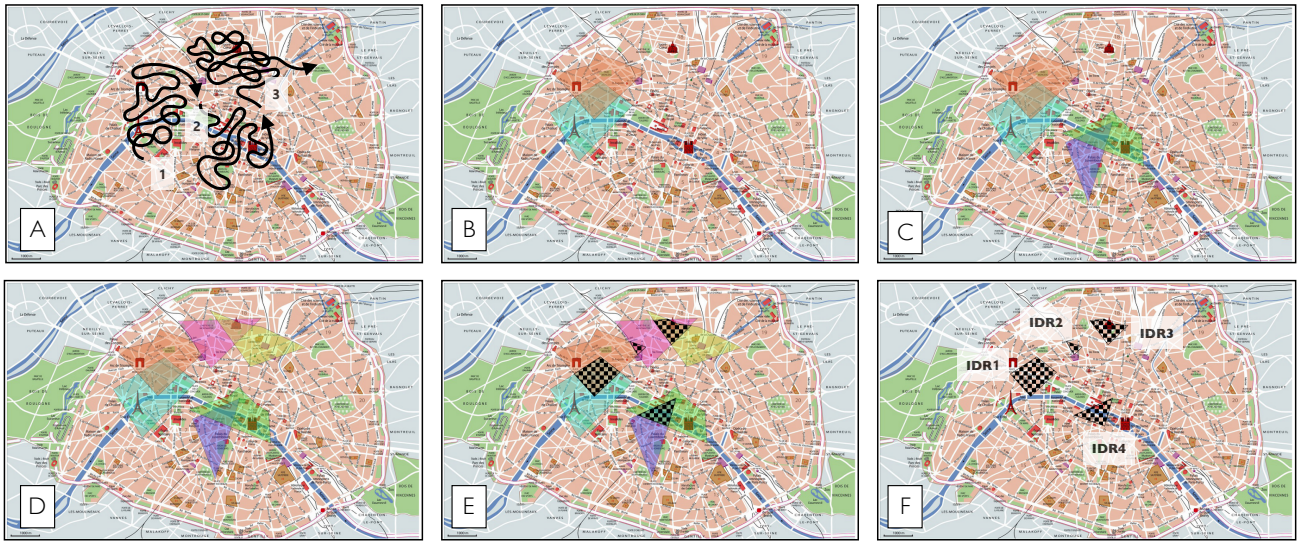

Figure 1: The process of finding IDRs on Airbnb dataset.

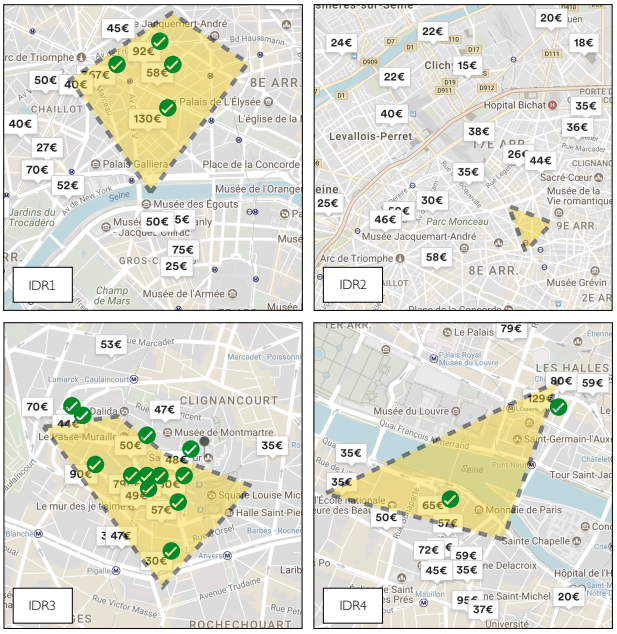

Figure 2: Matching points for IDR1 to IDR4.

consider $g = 3$ and capture Benício's feedback in three different time segments (progressing from Figures 1.B to 1.D). It shows that Benício started his search around Eiffel Tower and Arc de Triomphe (Figure 1.B) and gradually showed interest in south (Figure 1.C) and north (Figure 1.D) as well. All intersections between those clusters are discovered (hatching regions in Figure 1.E) which will constitute the set of IDRs (Figure 1.F), i.e., IDR1 to IDR4.

4.2 Matching Points

Being a function of mouse move points, IDRs are discovered in the interaction layer. We then need to find out which points in $\mathcal{P}$ fall into IDRs, hence forming the subset $\mathcal{P}_s$. We employ Equation 2 to transform those points from the spatial layer to the interaction layer. Then a simple "spatial containment" function can verify which points fit into the IDRs. Given a point p and an IDR r, a function $contains(p, r)$ returns "true" if p is inside r, otherwise "false". In our case, we simply use the implementation of $ST\_Within(p, r)$ module in PostGIS[5], i.e., our underlying spatial DBMS which hosts the data.

In the vanilla version of our spatial containment function, all points should be checked against all IDRs. Obviously, this depletes the execution time. To prevent the exhaustive scan, we employ Quadtrees [17] in a two-step approach.

- In an offline process, we build a Quadtree index for all points in $\mathcal{P}$. We record the membership relations of points and cells in the index.
- When IDRs are discovered, we record which cells in the Quadtree index intersect with IDRs. As we often end up with few IDRs, the intersection verification performs fast. Then for matching points, we only check a subset which is inside the cells associated to IDRs and ignore the points outside. This leads to a drastic pruning of points in $\mathcal{P}$.

We follow our running example and illustrate the matching process in Figure 2. In the Airbnb dataset, points are home-stays which are shown with their nightly price on the map. We observe that there exist many matching points with IDR3 and absolutely no matching point for IDR2. For IDR4, although there exist many home-stays below the region, we never check their containment, as they belong to a Quadtree cell which doesn't intersect with the IDR.

4.3 Updating Analyst Feedback Vector

The set of matching points $\mathcal{P}_s$ (line 2 of Algorithm 1) depicts the implicit preference of the analyst. We keep track of this preference in a feedback vector F. The vector is initialized by zero, i.e., the analyst has no preference at the beginning. We update F using the attributes of the points in $\mathcal{P}_s$.

We consider an *increment value* δ to update F. If $p \in \mathcal{P}_s$ gets v_1 for attribute a_1, we augment the value in the F's cell of $\langle a_1, v_1 \rangle$

[5] *https://postgis.net/docs/manual-dev/ST_Within.html*

Table 1: Attributes of points in IDR1.

ID	Price	#Beds	Balcony	Air-cond.	Rating
1	130€	1	Yes	Yes	5/5
2	58€	1	Yes	No	5/5
3	92€	2	Yes	No	5/5
4	67€	1	Yes	No	4/5

by δ. Note that we only consider incremental feedback, i.e., we never decrease a value in F.

We explain the process of updating the feedback vector using a toy example. Given the four matched points in IDR1 (Figure 2) with prices 130€, 58€, 92€ and 67€, we want to update the vector F given those points. Few attributes of these points are mentioned in Table 1. In practice, there are often more than 50 attributes for points. The cells of F are illustrated in the first column of Table 2. As three points get the value "1" for the attribute "#Beds", then the value in cell ⟨#Beds,1⟩ is augmented three times by δ. The same process is repeated for all attribute-values of points in $\mathcal{P}_s$. Note that all cells of F are not necessarily touched in the feedback update process. For instance, in the above example, 5 cells out of 12 remain unchanged.

By specifying an increment value, we can materialize the updates and normalize the vector using a Softmax function. We always normalize F in a way that all cell values sum up to 1.0. Given $\delta = 1.0$, the normalized values of the F vector is illustrated in the third column of Table 2. Higher values of δ increase the influence of feedbacks.

The normalized content of the vector F captures the implicit preferences of the analyst. For instance, the content of F after applying points in IDR1 shows that the analyst has a high interest in having a balcony in her home-stay, as her score for the cell ⟨Balcony,Yes⟩ is 0.25, i.e., the highest among other cells. This reflects the reality as all points in IDR1 has balcony. Note that although we only consider positive feedback, the Softmax function lowers the values of untouched cells once other cells get rewarded.

An important consideration in interpreting the vector F is that the value "0" does not mean the lowest preference, but *irrelevance*. For instance, consider the cell ⟨Rating,2⟩ in Table 2. The value "0" for this cell shows that the analyst has never expressed her implicit feedback on this facet. It is possible that in future iterations, the analyst shows interest in a 2-star home-stay (potentially thanks to its price), hence this cell gets a value greater than zero. However, cells with lower preferences are identifiable with non-zero values tending to zero. For instance, the value 0.06 for the cell ⟨Rating,4⟩ shows a lower preference towards 4-star home-stays compared to the ones with 5 stars, as only one point in $\mathcal{P}_s$ is rated 4 in IDR1.

4.4 Generating Highlights

The ultimate goal is to highlight k points to guide analysts in analyzing their spatial data. The updated feedback vector F is the input to the highlighting phase. We assume that points in IDRs are already investigated by the analyst. Hence our search space for highlighting is $\mathcal{P} - \mathcal{P}_s$.

We seek two properties in k highlights: *similarity* and *diversity*. First, highlights should be in the same direction of the analyst's implicit feedback, hence similar to the vector F. The similarity between a point $p \in \mathcal{P}$ and the vector F is defined as follows.

Table 2: Updating Analyst Feedback Vector

Attribute-value	Applying IDR 1	Normalized
⟨#Beds,1⟩	$+3\delta$	0.19
⟨#Beds,2⟩	$+\delta$	0.06
⟨#Beds,+2⟩	*(no update)*	0.00
⟨Balcony,Yes⟩	$+4\delta$	**0.25**
⟨Balcony,No⟩	*(no update)*	0.00
⟨Air-cond.,Yes⟩	$+\delta$	0.06
⟨Air-cond.,No⟩	$+3\delta$	0.19
⟨Rating,1⟩	*(no update)*	0.00
⟨Rating,2⟩	*(no update)*	0.00
⟨Rating,3⟩	*(no update)*	0.00
⟨Rating,4⟩	$+\delta$	**0.06**
⟨Rating,5⟩	$+3\delta$	0.19

$$similarity(p, F) = avg_{a \in \mathcal{A}}(sim(p, F, a)) \tag{3}$$

The *sim*() function can be any function such as Jaccard or Cosine. Each attribute can have its own similarity function (as string and integer attributes are compared differently.) Then *sim*() works as an overriding-function which provides encapsulated similarity computations for any type of attribute.

Second, highlighted points should also represent distinct directions so that the analyst can observe different aspects of data and decide based on the big picture. Given a set of points $\mathcal{P}_k = \{p_1, p_2 \dots p_k\} \subseteq \mathcal{P}$, we define *diversity* as follows.

$$diversity(\mathcal{P}_k) = avg_{\{p,p'\} \subset \mathcal{P}_k | p \neq p'} distance(p, p') \tag{4}$$

The function $distance(p, p')$ operates on geographical coordinates of p and p' and can be considered as any distance function of Minkowski distance family. However, as distance computations are done in the spherical space, a natural choice is to employ Haversine distance shown in Equation 5. Our application of diversity on geographical points differs from those of [13], because we consider geographical distance as the basis to calculate diversity between two points.

$$\begin{aligned} distance(p, p') = acos(&cos(p.lat) \times cos(p'.lat) \times cos(p.lon)) \\ &\times cos(p'.lon) + cos(p.lat) \times sin(p'.lat) \\ &\times cos(p.lon) \times sin(p'.lon) \\ &+ sin(p.lat) \times sin(p'.lat)) \\ &\times earth\_radius \end{aligned} \tag{5}$$

Algorithm 3 describes our approach for highlighting k similar and diverse points. We propose a best-effort greedy approach to efficiently compute highlighted points. We consider an offline step followed by the online execution of our algorithm.

In order to speed up the similarity computation in the online execution, we pre-compute an inverted index for each single point $p \in \mathcal{P}$ in the offline step (as is commonly done in the Web search). Each index $\mathcal{L}_p$ for the point p keeps all other points in $\mathcal{P}$ in decreasing order of their similarity with p.

The first step of Algorithm 3 is to find the most similar point to F, so-called p^*. The point p^* is the closest possible approximation of F in order to exploit pre-computed similarities. The algorithm makes sequential accesses to $\mathcal{L}_{p^*}$ (i.e., the inverted index of the point p^*) to greedily maximize diversity. Algorithm 3 does not sacrifice efficiency in price of value. We consider a *time limit*

Algorithm 3: Get k similar and diverse highlights *get_highlights*()

Input: Points $\mathcal{P}$, Feedback vector F, k, *time_limit*
Output: $\mathcal{P}_k$

```
1  p* ← max_sim_to(P, F)
2  P_k ← top_k(L_p*, k)
3  p_next ← get_next(L_p*)
4  while time_limit not exceeded do
5      for p_current ∈ P_k do
6          if diversity_improved(P_k, p_next, p_current) then
7              P_k ← replace(P_k, p_next, p_current)
8              break
9          end
10     end
11     p_next ← get_next(L_p*)
12 end
13 return P_k
```

parameter which determines when the algorithm should stop seeking maximized diversity. Scanning inverted indexes guarantees the similarity maximization even if time limit is chosen to be very restrictive. Our observations with several spatial datasets show that we achieve the diversity of more than 0.9 with time limit set to 200ms.

In line 2 of Algorithm 3, $\mathcal{P}_k$ is initialized with the k highest ranking points in $\mathcal{L}_{p^*}$. Function $get\_next(\mathcal{L}_{p^*})$ (line 3) returns the next point p_{next} in $\mathcal{L}_{p^*}$ in sequential order (as a common practice in information retrieval). Lines 4 to 12 iterate over the inverted indexes to determine if other points should be considered to increase diversity while staying within the time limit.

The algorithm looks for a candidate point $p_{current} \in \mathcal{P}_k$ to replace in order to increase diversity. The boolean function *diversity_improved*() (line 6) checks if by replacing $p_{current}$ by p_{next} in $\mathcal{P}_k$, the overall diversity of the new $\mathcal{P}_k$ increases. It is important to highlight that for each run of the algorithm, we only focus on one specific inverted list associated to the input point. Algorithm 3 verifies the similarity and diversity of each point with all other points, and then processes the normalization.

5 EXPERIMENTS

We discuss two sets of experiments. Our first set is on the usefulness of our approach. Then we focus more on discovering IDRs and present few statistics and insights for them.

First off, we validate the "usefulness" of our approach. For this aim, we design a user study with some participants who are all students of Computer Science. Some of them are "novice" users who don't know the location under investigation, and some are "experts." Participants should fulfill a task. For each participant, we report a variant of time-to-insight measure, i.e., how long the participants interact with the tool before fulfilling the task. Evidently, less number of interactions are preferred as it means that the participant can reach insights faster.

On the Airbnb[6] dataset of Paris with 1,000 points, we define two different types of tasks: *T1: "finding a point in a requested location"* (e.g., find a home-stay in the "*Champ de Mars*" area), and *T2: "finding a point with a requested profile"* (e.g., find a cheap home-stay.) Due to the vagueness associated to these tasks, participants require to go through an exploratory analysis session. Moreover, participants may also begin their navigation either from *I1: "close to the goal"* or *I2: "far from the goal"*.

Table 3: Interactions of "novice" and "expert" participants (in seconds)

	T1/I1	T2/I1	T1/I2	T2/I2
Novices	1.99	2.38	2.00	2.48
Experts	1.72	2.09	1.70	2.14

Table 4: IDR statistics on Airbnb dataset

# points	# regions	# IDRs	# points in IDRs	% points
100	11.35	10.05	29.40	29.40%
1000	10.75	6.75	11.70	1.17%
2000	7.37	3.63	5.63	0.003%
4000	10.30	10.15	53.15	1.33%
average	**9.94**	**7.64**	**25.97**	**8.05%**

Table 3 shows the results. We observe that on average it takes 2.067 seconds to achieve defined goals. This shows that implicit feedback capturing is an effective mechanism which helps analysts to reach their goals in a reasonable time. Expert participants need 0.35 seconds less time on average. Interestingly, starting points, i.e., *I1* and *I2*, do not have a huge impact on number of steps. It is potentially due to the diversity component which provides distinct options and can quickly guide analyst towards their region of interest. We also observe that the task *T2* is an easier task than *T1*, as on average it took less to be accomplished. This is potentially due to where the analyst can request options similar to what she has already observed and greedily move to her preferred regions.

In the second part of our experiments, we employ two different datasets, i.e., Airbnb and Yelp[7]. We pick a similar subset from both datasets, i.e., home-stays and restaurants in Paris city, respectively. We consider four different sizes of those datasets, i.e., 100, 1000, 2000 and 4000 points, respectively. For each size of the datasets, we manually perform 20 sessions, and then we present the results as the average of sessions.

We limit each session to 2 minutes where we seek for interesting points in the datasets. We capture the following information in each session:

- The number of regions created from the mouse moves during the session;
- The number of generated IDRs (intersection of regions);
- The number of points from the dataset presented in each IDR;
- The coverage of points (in the dataset) with IDRs collectively.

Tables 4 and 5 show the result for Airbnb and Yelp, respectively. In Table 4, we observe that the number of regions decreases when the number of points increases. On average, 10 regions are constructed per session. The average number of points presented in IDRs is 25.97. It shows that our approach is able to highlight at least 8.05% of points from the dataset, on average. We notice an outlier in the experiment with 2000 points in Tables 4. This happened due the fact that the analyst concentrated in a very small area generating a smaller number of IDRs, and consequently a smaller number of points.

[6] *http://insideairbnb.com/get-the-data.html*

[7] *https://www.yelp.com/dataset*

Table 5: IDR statistics on Yelp dataset

# points	# regions	# IDRs	# points in IDRs	% points
100	14.90	7.55	28.30	28.30%
1000	13.90	10.00	149.55	14.96%
2000	11.05	9.80	111.05	5.55%
4000	10.45	8.55	145.7	3.64%
average	**12.57**	**8.97**	**108.65**	**13.11%**

More uniform results are observed in Table 5, i.e., for Yelp dataset vis-à-vis Airbnb. The average number of generated regions reaches 12.75 per session. Also, the number of regions decreases by increasing the number of points. The same happens for IDRs, where we obtain an average of 8.9 IDRs generated per session. The number of points presented in IDRs is on average 108.65 and it represents on average 13.11% of points highlighted from the dataset.

6 RELATED WORK

To the best of our knowledge, the problem of spatial information highlighting using implicit feedback has not been addressed before in the literature. However, our work relates to few others in their semantics.

Information Highlighting. The literature contains few instances of information highlighting approaches [22, 27, 31, 32]. However, all these methods are objective, i.e., they assume that analyst's preferences are given as a constant input and will never change in the future. This limits their functionality for serving scenarios of exploratory analysis. The only way to fulfill "spatial guidance" is to consider the evolutionary and subjective nature of analyst's feedback. In our approach, the feedback vector gets updated in time based on the implicit feedback of the analyst.

Online recommendation approaches can also be considered as an information highlighting approach where recommended items count as highlights. Most recommendation algorithms are space-agnostic and do not take into account the spatial information. While few approaches focus on the spatial dimension [5, 13, 21], they still lack the evolutionary feedback capturing. Moreover, most recommendation methods miss "result diversification", i.e., highlights may not be effective due to overlaps.

Feedback Capturing. Several approaches are proposed in the state of the art for capturing different forms of feedback [8, 10, 12, 19, 25, 33]. The common approach is a top-k processing methodology in order to prune the search space based on the explicit feedback of the analyst and recommend a small subset of interesting results of size k. A clear distinction of our proposal is that it doesn't aim for pruning, but leveraging the actual data with potential interesting results (i.e., highlights) that the analyst may miss due to the huge volume of spatial data. Moreover, in a typical top-k processing algorithm, analyst's choices are limited to k. On the contrary, our IDR approach enables a freedom of choice where highlights get seamlessly updated with new analyst's choices.

Few works formulate fusing approaches of explicit and implicit feedbacks to better capture user preferences [1, 4, 24]. Our approach functions purely on implicit feedback and does not require any sort of explicit signal from the analyst.

Region Discovery. Our approach finds interesting dense regions (IDRs) in order to derive analyst's implicit preferences. There exist several approaches to infer a spatial region for a given set of points [2, 6, 7, 14, 16, 18]. The common approach is to cluster points in form of concave and convex polygons. In [7], an algorithm is proposed to verify if a given point p on the surface of a sphere is located inside, outside, or along the border of an arbitrary spherical polygon. In [14, 16], a non-convex polygon is constructed from a set of input points on a plane. In [2, 18], imprecise regions are delineated into a convex or concave polygon. In our approach, it is important to discover regions by capturing mouse move points. In case a concave polygon is constructed, the "dents" of such a polygon may entail points which are not necessarily in $\mathcal{M}$. In the IDR's algorithm, however, we adapt Quickhull [6], due its simplicity, efficiency and it's natural implementation of convex polygons.

7 LIMITATIONS

In this paper, we presented a solution for highlighting out-of-sight information using a polygon-based approach for capturing implicit feedbacks. To the best of our knowledge, our work is the first effort towards formalizing and implementing information highlighting using implicit feedback. However, we consider our work as an on-going effort where we envision to address some limitations in the future, such as "customizability", "performance", "cold start", and "quantitative experiments".

In this section we present some limitations of our proposed work, describing what we will consider as future work. One limitation is about the "customizable" use of geographical maps as an interaction means. As we only consider static maps, we plan to work on translations and rotations as a future work. Another gap that we envision to work on is performance. We plan to run an extensive performance study to detect bottlenecks of our approach.

Our problem bears similarities with recommendation algorithms where the quality of the output may be influenced by scarce availability of input. This problem is referred to as the cold start problem [20]. While there is no guarantee for a meaningful highlight in case of the complete absence of implicit feedbacks, our approach can return a reasonable set of highlights even with one single iteration of mouse moves. In the future, we envision to tackle the no-input challenge by leveraging statistical properties of the spatial data to obtain a default view for highlights.

Another limitation is the medium-size datasets to be processed. Our algorithm processes similarity and diversity in an $O(n^2)$ complexity. Also Quickhull [6] uses a divide and conquer approach similar to that of Quicksort, and its worst complexity is $O(n^2)$. While processing a 10K-point dataset is straightforward in our framework, we plan to experiment with larger datasets in the future by improving our algorithms towards better performance. Another direction for future work is to consider experiments which measure the quantitative and qualitative influence of each component separately.

8 CONCLUSION

In this paper, we present an approach to explore Interesting Dense Regions (IDRs) using implicit feedback in order to detect analyst latent preferences. The implicit feedbacks are captured from mouse moves of analysts over the geographical map while analyzing spatial data. We formalize a novel polygon-based mining algorithm which returns few highlights in-line with analyst's implicit preferences. The highlights enable analysts to focus on what matters the most and prevent information overload.

We consider various future directions for this work. First, we are interested to incorporate an "explainability" component which can describe causalities behind preferences. For instance, we are interested to find seasonal patterns to see why the preferences of analysts change from place to place during various seasons of the year. Another direction is to incorporate "Query by Visualization" approaches, where analysts can specify their intents alongside their implicit preferences, directly on the map [29].

REFERENCES

[1] Eoin Mac Aoidh, Michela Bertolotto, and David C. Wilson. 2007. Analysis of implicit interest indicators for spatial data. In *15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings*. 47. https://doi.org/10.1145/1341012.1341071

[2] Avi Arampatzis, Marc van Kreveld, Iris Reinbacher, Christopher B. Jones, Subodh Vaid, Paul Clough, Hideo Joho, and Mark Sanderson. 2006. Web-based delineation of imprecise regions. *Computers, Environment and Urban Systems* 30, 4 (2006), 436 – 459. https://doi.org/10.1016/j.compenvurbsys.2005.08.001 Geographic Information Retrieval (GIR).

[3] Ioannis Arapakis, Mounia Lalmas, and George Valkanas. 2014. Understanding Within-Content Engagement Through Pattern Analysis of Mouse Gestures. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. ACM, New York, NY, USA, 1439–1448. https://doi.org/10.1145/2661829.2661909

[4] Andrea Ballatore and Michela Bertolotto. 2011. Semantically Enriching VGI in Support of Implicit Feedback Analysis. In *Web and Wireless Geographical Information Systems*, Katsumi Tanaka, Peter Fröhlich, and Kyoung-Sook Kim (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 78–93.

[5] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in location-based social networks: a survey. *GeoInformatica* 19, 3 (2015), 525–565. https://doi.org/10.1007/s10707-014-0220-8

[6] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The Quickhull Algorithm for Convex Hulls. *ACM Trans. Math. Softw.* 22, 4 (Dec. 1996), 469–483. https://doi.org/10.1145/235815.235821

[7] Michael Bevis and Jean-Luc Chatelain. 1989. Locating a point on a spherical surface relative to a spherical polygon of arbitrary shape. *Mathematical Geology* 21, 8 (01 Oct 1989), 811–828. https://doi.org/10.1007/BF00894449

[8] Mansurul Bhuiyan, Snehasis Mukhopadhyay, and Mohammad Al Hasan. 2012. Interactive pattern mining on hidden data: a sampling-based solution. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 95–104.

[9] Derya Birant and Alp Kut. 2007. ST-DBSCAN: An Algorithm for Clustering Spatial-temporal Data. *Data Knowl. Eng.* 60, 1 (Jan. 2007), 208–221. https://doi.org/10.1016/j.datak.2006.01.013

[10] Mario Boley, Michael Mampaey, Bo Kang, Pavel Tokmakov, and Stefan Wrobel. 2013. One click mining: Interactive local pattern discovery through implicit preference and performance learning. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*. ACM, 27–35.

[11] Mon Chu Chen, John R. Anderson, and Myeong Ho Sohn. 2001. What Can a Mouse Cursor Tell Us More?: Correlation of Eye/Mouse Movements on Web Browsing. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems (CHI EA '01)*. ACM, New York, NY, USA, 281–282. https://doi.org/10.1145/634067.634234

[12] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2016. AIDE: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering* 28, 11 (2016), 2842–2856.

[13] Marina Drosou and Evaggelia Pitoura. 2012. DisC diversity: result diversification based on dissimilarity and coverage. *PVLDB* 6, 1 (2012), 13–24. http://dl.acm.org/citation.cfm?id=2428538

[14] Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. 2008. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition* 41, 10 (2008), 3224 – 3236. https://doi.org/10.1016/j.patcog.2008.03.023

[15] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 226–231. http://dl.acm.org/citation.cfm?id=3001460.3001507

[16] M.J. Fadili, M. Melkemi, and A. ElMoataz. 2004. Non-convex onion-peeling using a shape hull algorithm. *Pattern Recognition Letters* 25, 14 (2004), 1577 – 1585. https://doi.org/10.1016/j.patrec.2004.05.015

[17] Raphael A. Finkel and Jon Louis Bentley. 1974. Quad trees a data structure for retrieval on composite keys. *Acta informatica* 4, 1 (1974), 1–9.

[18] Antony Galton and Matt Duckham. 2006. What Is the Region Occupied by a Set of Points?. In *Geographic Information Science*, Martin Raubal, Harvey J. Miller, Andrew U. Frank, and Michael F. Goodchild (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 81–98.

[19] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. 2014. Distributed and interactive cube exploration. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE, 472–483.

[20] Vincent Leroy, Berkant Barla Cambazoglu, and Francesco Bonchi. 2010. Cold start link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*. 393–402. https://doi.org/10.1145/1835804.1835855

[21] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. 2012. LARS: A Location-Aware Recommender System. In *ICDE*. 450–461. https://doi.org/10.1109/ICDE.2012.54

[22] J. Liang and M. L. Huang. 2010. Highlighting in Information Visualization: A Survey. In *2010 14th International Conference Information Visualisation*. https://doi.org/10.1109/IV.2010.21

[23] Lauro Lins, James T Klosowski, and Carlos Scheidegger. 2013. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2456–2465.

[24] Nathan N. Liu, Evan W. Xiang, Min Zhao, and Qiang Yang. 2010. Unifying Explicit and Implicit Feedback for Collaborative Filtering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, New York, NY, USA, 1445–1448. https://doi.org/10.1145/1871437.1871643

[25] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Alexandre Termier. 2015. Interactive User Group Analysis. In *CIKM*. ACM, 403–412. https://doi.org/10.1145/2806416.2806519

[26] Behrooz Omidvar-Tehrani, Plácido A Souza Neto, Felipe M Freire Pontes, and Francisco Bento. 2017. GeoGuide: An Interactive Guidance Approach for Spatial Data. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on*. IEEE, 1112–1117.

[27] Anthony C. Robinson. 2011. Highlighting in Geovisualization. *Cartography and Geographic Information Science* 38, 4 (2011), 373–383. https://doi.org/10.1559/15230406384373 arXiv:http://dx.doi.org/10.1559/15230406384373

[28] John F. Roddick, Max J. Egenhofer, Erik G. Hoel, Dimitris Papadias, and Betty Salzberg. 2004. Spatial, Temporal and Spatio-Temporal Databases - Hot Issues and Directions for PhD Research. *SIGMOD Record* 33, 2 (2004), 126–131. https://doi.org/10.1145/1024694.1024724

[29] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless data exploration with zenvisage: an expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment* 10, 4 (2016), 457–468.

[30] Aditya Telang, Deepak Padmanabhan, and Prasad Deshpande. 2012. Spatio-temporal Indexing: Current Scenario, Challenges and Approaches. In *Proceedings of the 18th International Conference on Management of Data (COMAD '12)*. Computer Society of India, Mumbai, India, India, 9–11. http://dl.acm.org/citation.cfm?id=2694443.2694449

[31] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. 2007. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1129–1136.

[32] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *TVCG* 22, 1 (2016).

[33] Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. 2006. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 773–778.

[34] Jia Yu, Zongsi Zhang, and Mohamed Sarwat. 2018. Spatial data management in apache spark: the GeoSpark perspective and beyond. *GeoInformatica* (2018), 1–42.