SEQ: Example-based Query for Spatial Objects

ABSTRACT

Spatial object search is prevalent in map services (e.g., Google Maps). To rent an apartment, for example, one will take into account its nearby facilities, such as *supermarkets*, *hospitals*, and *subway stations*. Traditional keyword search solutions, such as the *nearby* function in Google Maps, are insufficient in expressing the often complex attribute/spatial requirements of users. Those requirements, however, are essential to reflect the user search intention. In this paper, we propose the Spatial Exemplar Query (SEQ), which allows the user to input a result example over an interface inside the map service. We then propose an effective similarity measure to evaluate the proximity between a candidate answer and the given example. We conduct a user study to validate the effectiveness of SEQ. Our result shows that more than 88% of users would like to have an example assisted search in map services. Moreover, SEQ gets a user satisfactory score of 4.3/5.0, which is more than 2 times higher than that of a baseline solution.

1 INTRODUCTION

Searching spatial objects, which often takes multiple objects (i.e.,their attributes) and the relative distances among them into consideration, is prevalent in POI (Point of Interest) recommendation, trip planning, and geo-social analytics. An typical scenario in trip planning is that tourists are interested in a set of co-located hotels and attractions, wishing to find a hotel with a good *rating* (attribute 1), a low *cost* (attribute 2) and proper distances to *attractions* (other objects).

Notably, different users have various personal requirements on relative distances of their interesting objects. To find a house, for example, one often considers neighboring educational resources and access to daily facilities such as supermarkets, hospitals and subway stations. Parents with young children favor the house to be surrounded by abundant school resources for the education purpose, while other elder couples may care more about the easy access to supermarkets or other daily facilities. As such, the relative distances of objects interested by these two kinds of people can be different. As another example, in geo-social analytics, crime analysts are interested in analyzing the spatial layout of a series of criminal spots (e.g., robbery series) [2]. Different layouts of criminal spots, e.g., the spots compositing in a line or a circle, are used in analyzing different crimes [1].

Traditional keyword search solutions, such as the *nearby* function [1] in Google Maps, is not capable of capturing users' complex search intention (e.g., object attribute requirements, and distance requirements among objects), as it only allows a user to search *a single type* of interesting objects that are close to a *specific* place. Another way to express such complicated requirements is to use SQL queries to indicate specific constraints such as distance thresholds and attribute constraints. However, it is difficult for non-expert users to write sophisticated SQL queries.

[1]https://goo.gl/3ibV5z

CIKM'17, Singapore
© 2017 ACM. 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

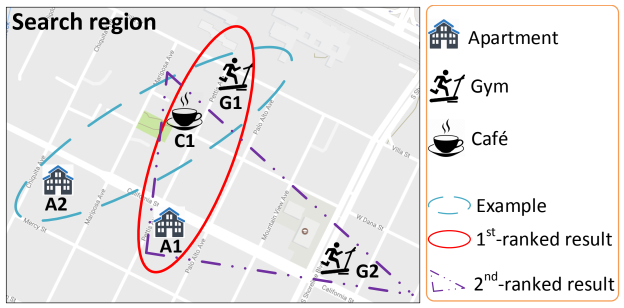


Figure 1: An example SEQ.

To assist a user effectively express her search intention, we propose an example-based solution called the SEQ (Spatial Exemplar Query). In SEQ, users can use an *example* to describe their requirements on the objects. The example is a group of objects that carry the spatial proximity information (e.g, the relative locations of objects) and attribute requirements. Users can have several ways to find a suitable example as the input to SEQ. For instance, a user can navigate the online map to mark an example [2]; alternatively, the user can first input relevant keywords, e.g., *apartment, gym, cafe*, to seek for candidate combinations [3], then pick a combination of objects as the example that best fits the user's search intention. Given the example, a search region and an integer k, SEQ first automatically generates an internal pattern of the given example, including its attribute values and relative locations of objects. Then, SEQ searches within the region to retrieve k groups of objects that are the most similar to the given example based on their internal patterns.

We illustrate an example of SEQ in Fig. 1: a sports-lover considers renting an apartment that is close to a gym, while she also wants to have some coffee after doing some physical exercises. In this scenario, she is interested in the combination (apartment, gym, cafe) such that the cafe is on the road from the gym to the apartment (i.e., three objects forming a route). She navigates the online map to find a good example $(A2, C1, G1)$. With this example and a search region as the input to SEQ, SEQ outputs the top-k (with $k = 2$) results according to their similarity to the example. Intuitively, the 1^{st}-ranked result fits best to the user requirement, as in this solution she can easily find a cafe when she goes back to the apartment $(A1)$ from the gym $(G1)$. However, such a typical user requirement is hard to describe with traditional location search services, e.g., the *nearby* function in Google Maps. In particular, the *nearby* function cannot distinguish the 2 results very well because it outputs the results as

[2]A user interaction interface is needed to implement this function.
[3]This function is not in Google Maps, but should be easy to support.

long as the objects are within a specified range. This is unsatisfactory. For example, if the 2^{nd}-ranked result is recommended, then the user needs a long detour to the cafe from the gym before she goes back to the apartment, in order to have a coffee after doing her exercises.

Our contributions. To our best knowledge, our work is the first to integrate a query-by-example paradigm into the search for a group of spatial objects. In particular, we motivate the study of Spatial Exemplar Query (SEQ), which can relieve the burden for the user to express her search intention. Second, we define a novel similarity measure and use it to find results that are the k most similar to the example. Observing that a simple solution often costs more than 2 minutes to evaluate the SEQ, we propose an efficient method to reduce the cost to less than 1 second. Finally, we conduct experiments and a user study on real datasets to validate the effectiveness and efficiency of our methods.

2 RELATED WORK

QBE in Relational Database. Query by example (in short, QBE) [4, 6–8] is recently studied in relational database. QBE allows users to input a number of desired sample output tuples. Then, QBE returns to the user one SQL whose execution generates those examples as output. Existing studies of QBE [4, 6–8] aim at recovering SQL queries by using *a large set of result tuples*. Hence, they are difficult to apply in map services. The reason is that it is infeasible to require a user to input a large set of examples in map services.

Spatial-keyword Queries. Spatial-keyword query (e.g., [3, 5, 9]) returns objects that are both relevant to the input textual information and spatial information. The focus of [3, 5, 9] is the efficient evaluation of a query with rigid semantics, while our focus is to simplify query expressions in the spatial search. Our example-based method can complement their works in enhancing the expression of users' intention. We also conduct a user study to compare our method with a typical spatial-keyword query named Range Keyword Query [5]. The results show that our experiments are more preferred by the users.

3 DATA MODEL AND FORMULATIONS

3.1 Data model

A spatial network contains a collection of objects O. Each object $p \in O$ is associated with a type T_p, a list of attributes A_p, and a coordinate (lat_p, lon_p). For example, the type of a Starbucks store is a *coffee store*. Its attribute can be the *customer rating*, which indicates the customer impression on the store [4]. We define a *tuple* $t = \{p_1, \ldots, p_m\}$ as a set of ordered objects, and use $t[i]$ to represent p_i. Two tuples t_1 and t_2 are said to have the same category, if for any $i \in [1, m]$, their i^{th} objects have the same type.

3.2 Problem Formulation

A natural way to define the spatial structure of objects inside tuple t is to define the vector containing pairwise Euclidean distances [5] between two objects of tuple t, as follows.

[4]the values are normalized to [0,1].

[5]Our solution can easily incorporate other measures such as road network distances.

Definition 3.1 (Distance Vector). The distance vector V_t for a tuple t=$\{p_1, \ldots, p_m\}$, is the vector V_t=$\{$dist(p_1, p_2), dist$(p_1, p_3), \ldots,$ dist(p_1, p_m), dist$(p_2, p_3), \ldots,$ dist$(p_2, p_m), \ldots,$ dist$(p_{m-1}, p_m)\}$.

The dimension of the distance vector V_t is $m' = \frac{m(m-1)}{2}$. By the definition of the distance vector, it is then natural to define the *spatial similarity* of two tuples as the cosine similarity of their corresponding distance vectors, as follows.

Definition 3.2 (SIM$_s$). The spatial similarity between two tuples t_1 and t_2 is $\text{SIM}_s(t_1, t_2) = cos(V_{t_1}, V_{t_2})$.

Similarly, to measure the *attribute similarity*, we define the *Attribute Vector* of an object.

Definition 3.3 (Attribute Vector). The attribute vector U_t for an object p with attributes $A_p = \{a_1, \ldots, a_h\}$, is the vector $U_t = \{u_1, \ldots, u_h\}$, such that u_i is the value of attribute a_i for tuple t.

Then, the attribute similarity between two tuples is naturally measured by the average cosine similarity between their corresponding objects. We therefore define the following SIM$_a$ to measure it.

Definition 3.4 (SIM$_a$). The attribute similarity between two size-m tuples t_1 and t_2 that are in the same category, is $\text{SIM}_a(t_1, t_2) = \frac{1}{m}\sum_{i=1}^{m} cos(U_{t_1[i]}, U_{t_2[i]})$.

Given spatial similarity and attribute similarity of two tuples, we combine them to define the *Tuple Similarity* between tuple t_1 and t_2, SIM(t_1, t_2).

Definition 3.5 (SIM). For two tuples that are in the same category, their tuple similarity is $\text{SIM}(t_1, t_2)$=$\alpha\cdot$ $\text{SIM}_s(t_1, t_2)$ +$(1-\alpha)\cdot$ $\text{SIM}_a(t_1, t_2)$, with respect to parameter $\alpha \in [0, 1]$.

Based on Definition 3.5, we define the SEQ problem as follows.

Definition 3.6 (SEQ). Given a spatial range R, an integer k, an example tuple t^*. The Spatial Exemplar Query (SEQ) returns top-k similar tuples $t_1, \cdots, t_k$ to t^* with respect to the tuple similarity, such that all objects in t_i $(i \in [1, k])$ locate in R, and t_i $(i \in [1, k])$ has the same category as t^*.

Let us go back to the example in Fig. 1 to illustrate SEQ. The input example tuple contains one apartment, one cafe and one gym. The top-2 results returned are ordered by their tuple similarity (i.e., SIM) to the input example tuple.

4 EFFICIENT TOP-K SEARCH

A straightforward approach to address SEQ is to enumerate all candidate tuples (hereinafter referred to as *candidates*) that have the same category as the example tuple t^* in region R, and select k candidates that are the most similar to t^*. However, the number of such candidates can be very large. For example, there can be thousands of objects whose types are "apartment" and "restaurant". Then there will be millions of combinations of (apartment, restaurant) as candidates. We therefore derive two effective theoretical bounds on SIM$_a$ and SIM$_s$ to prune a large number of candidates. With our method, our experiments show that more than 96% candidates are pruned.

Our idea is to examine the candidates in a certain order to enable effective candidate pruning. For ease of presentation, we first introduce the *rank presentation* of a candidate. Let $list_i$ $(i \in [1, m])$ be the list containing all the objects that have the same type as $t^*[i]$ (i.e., the i^{th} object in t^*). We also suppose that those objects in $list_i$ are sorted in a descending order of their attribute similarity to the object $t^*[i]$ (i.e., sorting object $o \in list_i$ according to $cos(U_o, U_{t^*[i]})$). If an object appears in the j^{th} position of the sorted list, we say its *rank* is j. Now, with the lists ($list_1$ to $list_m$), a candidate is a tuple containing one object from each of the lists. Hence every candidate $s = \{p_1, \ldots, p_m\}$ can be represented by the ranks of each of its elements. We denote the representation as $R(s)$. For example, a candidate $R(s) = \{r_1, \ldots, r_m\}$ means object p_i ranks r_i in $list_i$.

We observe that many candidates share a certain *prefix* (i.e., the first few elements). Hence, if we can determine an upper bound on $\text{SIM}(s, t^*)$ for candidate s *based only on a certain prefix of* s, and the upper bound is less than the k^{th} largest similarity to t^*, then we can safely *prune all candidates that share the same prefix.* The reason is, the similarity between t^* and such candidates with the same prefix must not exceed the upper bound, and thus such candidates are worse than the current k best candidates that have been inspected. Next, we derive upper bounds respectively for SIM_a and SIM_s, which can be combined to derive an upper bound for SIM.

Bounding SIM_a based on Prefix. Suppose there are two tuples s and s^* of size m, whose rank representations are $R(s) = \{r_1, \ldots, r_i, r_{i+1}, \ldots, r_m\}$ and $R(s^*) = \{r_1, \ldots, r_i, 1, \ldots, 1\}$. Then, $\text{SIM}_a(s, t^*) \leq \text{SIM}_a(s^*, t^*)$ since each list is sorted according to SIM_a. That means, for any candidate s, we can use the value $\text{SIM}_a(s^*, t^*)$ as an upper bound of $\text{SIM}_a(s, t^*)$, where s^* shares a certain prefix of s and the remaining objects in s^* all rank first in the corresponding sorted list.

Bounding SIM_s based on Prefix. For any tuple s, given its length-i prefix (e.g., $(s[1], \ldots, s[i])$), we can determine part of its corresponding distance vector $V_s = \{d_1, \ldots, d_u, ?, \ldots, ?\}$, where $u = \frac{i(i-1)}{2}$ and ? refers to some unknown value. We use Lemma 1 to bound $\text{SIM}_s(s, t^*)$ based on the prefix of s. Note that in Lemma 1, x'_j can be computed before examining any candidate, as $\{x'_1, \ldots, x'_{m'}\}$ is the normalized distance vector of the example tuple t^*. Hence, when a prefix of a candidate is known (e.g., y_1 to y_u are known), the values of A, C and $\sum_{j=u+1}^{m'} x'^2_j$ can be easily computed, and thus the upper bound value can be evaluated.

LEMMA 1. *For a size-m tuple s, let* $m' = \frac{m(m-1)}{2}$, $V_{t^*} = \{x_1, \ldots, x_{m'}\}$ *and* $V_s = \{y_1, \ldots, y_{m'}\}$. *Then* $\forall u \in [1, m')$, *we have* $SIM_s(s, t^*) \leq \sqrt{\frac{A^2}{C} + \sum_{j=u+1}^{m'} x'^2_j}$, *where* $x'_j = \frac{x_j}{\sqrt{\sum_{j=1}^{m'} x^2_j}}$, $A = \sum_{j=1}^{u} x'_j y_j$, $C = \sum_{j=1}^{u} y^2_j$.

Proof. Let $D = \sum_{i=u+1}^{m'} y^2_i$, then we have

$$\begin{aligned}
\text{SIM}_s(s, t^*) = cos(V_s, V_{t^*}) &= \frac{A + \sum_{j=u+1}^{m'} x'_j y_j}{\sqrt{C+D}} \\
&= \frac{A}{\sqrt{C+D}} + \frac{1}{\sqrt{C+D}} \sum_{j=u+1}^{m'} x'_j \cdot y_j \\
(\text{Cauchy Inequality }^{6}) &\leq \frac{A}{\sqrt{C+D}} + \frac{1}{\sqrt{C+D}} \sqrt{\sum_{j=u+1}^{m'} x'^2_j \cdot \sum_{j=u+1}^{m'} y^2_j} \\
&= \frac{A}{\sqrt{C+D}} + \frac{1}{\sqrt{C+D}} \sqrt{\sum_{j=u+1}^{m'} x'^2_j \cdot D}
\end{aligned} \tag{1}$$

Let $w = \frac{D}{C}$, then

$$\begin{aligned}
&\frac{A}{\sqrt{C+D}} + \frac{1}{\sqrt{C+D}} \sqrt{\sum_{j=u+1}^{m'} x'^2_j \cdot D} \\
&= \frac{A}{\sqrt{C}} \cdot \frac{1}{\sqrt{1+w}} + \sqrt{\sum_{j=u+1}^{m'} x'^2_j} \cdot \sqrt{\frac{w}{w+1}} \\
(\text{Cauchy Inequality}) &\leq \sqrt{\left(\left(\frac{A}{\sqrt{C}}\right)^2 + \left(\sqrt{\sum_{j=u+1}^{m'} x'^2_j}\right)^2\right) \cdot \left(\left(\frac{1}{\sqrt{1+w}}\right)^2 + \left(\sqrt{\frac{w}{w+1}}\right)^2\right)} \\
&= \sqrt{\frac{A^2}{C} + \sum_{j=u+1}^{m'} x'^2_j}
\end{aligned} \tag{2}$$

Depth-First Search for Pruning. For any candidate with a prefix $\{s[1], \ldots, s[i]\}$, we can combine the aforementioned two bounds to derive an upper bound on $\text{SIM}(s, t^*)$. To make use of such an upper bound, we examine the candidates in a *Depth-First-Search* manner. Formally, candidate s_1 is examined before candidate s_2, if and only if there exists j such that $s_1[i] = s_2[i]$ ($\forall 1 \leq i < j$) and $s_1[j]$ ranks higher than $s_2[j]$ in $list_j$. In this order, candidates that share long prefixes are clustered together. If we find that the prefix-derived upper bound is less than the current k^{th} largest similarity to t^*, we can skip the examination of all those candidates that share the prefix. Due to space limitations, we omit the pesudocode. We give an example as follows.

Example. Let $list_1 = \{u_1, u_2\}$, $list_2 = \{v_1\}$, $list_3 = \{w_1, w_2\}$. Then, the DFS search visits the candidates in the order: $\{u_1, v_1, w_1\}$, $\{u_1, v_1, w_2\}$,$\{u_2, v_1, w_1\}$,$\{u_2, v_1, w_2\}$. If the upper bound based on the prefix $\{u_2, v_1\}$ is less than the current k^{th} largest similarity, we can skip the examination of the last two candidates.

5 EVALUATION

Dataset and Queries. We use the POI dataset published by Yelp [7], including 77,444 POIs. For each POI, its geo location and attributes, e.g., ratings and categories are all recorded. We randomly create 100 queries within a circular range of radius r. The tuple size is 3 by default. Table 1 summarizes the main parameters used in the experiments (default values are in bold).

Algorithms. We implemented `BASIC` (the direct enumeration solution) and `PRUNE` (the algorithm with pruning) in C++. Both of them are introduced in Sec. 4. Our experiments are conducted on a machine with a 2.3 GHz Intel Core i7 processor and 16GB memory.

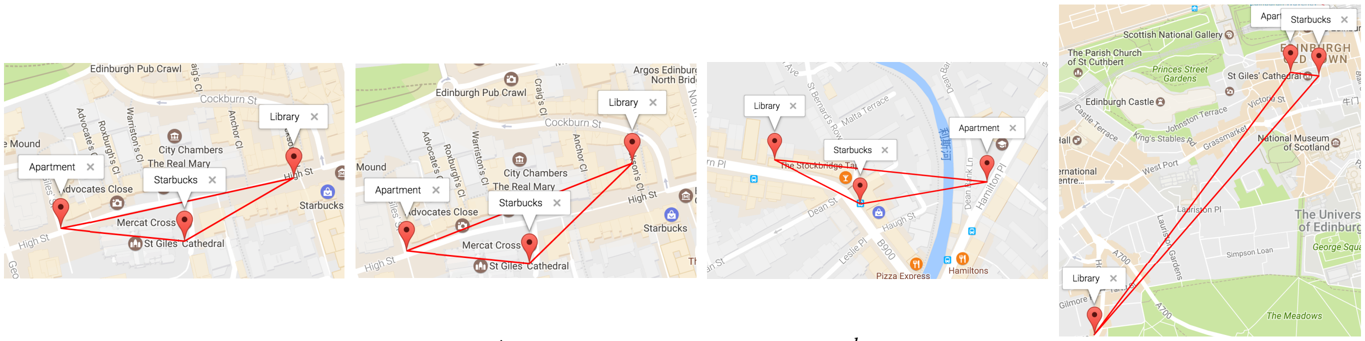
Param	Description	Value
α	Relative weights of the two similarities	0.1, 0.3, **0.5**, 0.7, 0.9
r	Valid search range radius	1, 2, **3**, 4, 5 (km)
k	The number of returned results	1, **5**, 10, 20, 50

Table 1: Experiment parameters

Efficiency. The average running times of `PRUNE` are shown in Table 2. The performance is not sensitive to the changes of α because both SIM_a and SIM_s need to be calculated as long as $\alpha > 0$. When we increase k or the search range r, the running time increases, since more candidates will be considered when k (or r) increases.

[6] $\sum_{i=1}^{m} x_i y_i \leq \sqrt{\sum_{i=1}^{m} x^2_i \sum_{i=1}^{m} y^2_i}$

[7] https://www.yelp.com/dataset_challenge



(a) The given example (b) The 1^{st}-ranked result (c) The 2^{nd}-ranked result (d) Result of RKS [5]

Figure 2: Case study of SEQ.

α	0.1	0.3	0.5	0.7	0.9
Time (ms)	316	336	338	349	380
Pruning effectiveness (%)	98.8	98.7	98.5	98.1	97.1
k	1	5	10	20	50
Time (ms)	32	338	349	363	403
Pruning effectiveness (%)	99.8	98.5	98.1	97.5	96.5
r	1	2	3	4	5
Time (ms)	68	161	338	586	847
Pruning effectiveness (%)	97.6	98.3	98.5	99.5	99.5

Table 2: Efficiency evaluation

However, even when k (or r) is set to a large value, `PRUNE` still performs within 1 second for each query. The main reason is that a large number of candidates are pruned by the proposed effective bounding techniques. As shown in Table 2, the *Pruning effectiveness* is always more than 96%, which means more than 96% candidates are pruned. We omit the running time of `BASIC`, as on average `BASIC` cannot finish a query within 2 minutes, which is infeasible in practice.

Case Study. Fig. 2 demonstrates an intuitive example of SEQ and its comparison with the Range Keyword Search (RKS) [5], which generalizes the Google Map *nearby* function to multiple types of objects [8]. Fig. 2(a) is the input example; Figs. 2(b) and (c) show the top-2 results. The 1^{st}-ranked result is located in almost the same region as of the given example, while 2^{nd}-ranked result is a set of objects which are relatively distant from the objects in the example. The distances between the objects, in both results, are quite similar to the example. In addition, the result indicates that the definition of SEQ is able to capture the spatial pattern (the relative locations and directions of objects). In contrast, the result given by RKS, as shown in Fig. 2(d), does not preserve the spatial pattern of the input example. Intuitively, we think this result is not as good as the SEQ results.

User Study. To validate the usefulness of SEQ, we conducted a user study. There are two goals. First, we want to know *whether users will choose to use examples, complementing to keywords, to express their search interest*; Second, when compared with the traditional methods like RKS, *whether SEQ returns better results*. We invited 26 university students to participate in this study. At the first of the user study, we briefly presented two definitions of SEQ and RKS, and asked students an open question whether they would choose to use SEQ. 88% of students answered they would alternatively choose SEQ. Then, we presented 16 pairs of comparisons between our result and the result of RKS for some typical search intentions. For example, a tourist is looking for a low-cost hotel that is close to some attractions. In each comparison, they need to rate their satisfaction about the two results (assuming they are issuing the search intention themselves). In addition, for the $416 (= 26 \times 16)$ queries tested in the study, our SEQ results get an average of $4.3/5.0$ satisfactory score, which is significantly better than that of RKS, whose average score is $1.8/5.0$.

[8]Current Google Maps *nearby* function only allows the user to select one place, and search one type of objects nearby.

6 CONCLUSIONS AND FUTURE WORK

We propose SEQ that assists users in expressing their search intention in map services. We believe that SEQ lies in the intersection of research fields such as information retrieval, human-computer interaction and NLP. This topic can be extended in several interesting ways.

First, it is interesting to integrate SEQ into a real map service, including designing a user-friendly interface, and studying how SEQ can complement existing functions in map services. Another direction is to learn the search intention of users by conducting SEQ iteratively in a user-system interactive manner. It is also possible to provide users with multiple ways to create their examples. For instance, users can draw a sketch, or use natural language to describe their desired examples.

REFERENCES

[1] http://www.iaca.net/Resources/Articles/identifyingcrimepatterns.pdf, 2002.
[2] http://www.iaca.net/Training/Resources/CrimeStatIIIWorkbook.pdf, 2002.
[3] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *VLDB*, 2(1):337–348, 2009.
[4] H. Li, C. Chan, and D. Maier. Query from examples: An iterative, data-driven approach to query construction. *VLDB*, 8(13):2158–2169, 2015.
[5] S. Luo, Y. Luo, S. Zhou, G. Cong, J. Guan, and Z. Yong. Distributed spatial keyword querying on road networks. In *EDBT*, pages 235–246, 2014.
[6] F. Psallidas, B. Ding, K. Chakrabarti, and S. Chaudhuri. S4: top-k spreadsheet-style search for query discovery. In *SIGMOD*, pages 2001–2016, 2015.
[7] Q. T. Tran, C. Chan, and S. Parthasarathy. Query by output. In *SIGMOD*, pages 535–548, 2009.
[8] Q. T. Tran, C. Y. Chan, and S. Parthasarathy. Query reverse engineering. *VLDB J.*, 23(5):721–746, 2014.
[9] D. Zhang, C.-Y. Chan, and K.-L. Tan. Processing spatial keyword query as a top-k aggregation query. In *SIGIR*, pages 355–364, 2014.