2017 IEEE 33rd International Conference on Data Engineering

Direction-Aware Why-Not Spatial Keyword Top-k Queries

Lei Chen*, Yafei Li†*, Jianliang Xu*, Christian S. Jensen‡
*Department of Computer Science, Hong Kong Baptist University, Hong Kong
†School of Information Engineering, Zhengzhou University, Zhengzhou, China
‡Department of Computer Science, Aalborg University, Denmark
{lchen, yafeili,xujl}@comp.hkbu.edu.hk; csj@cs.aau.dk

***Abstract*—With the continued proliferation of location-based services, a growing number of web-accessible data objects are geo-tagged and have text descriptions. An important query over such web objects is the *direction-aware spatial keyword query* that aims to retrieve the top-k objects that best match query parameters in terms of spatial distance and textual similarity in a given query direction. In some cases, it can be difficult for users to specify appropriate query parameters. After getting a query result, users may find some desired objects are unexpectedly missing and may therefore question the entire result. Enabling why-not questions in this setting may aid users to retrieve better results, thus improving the overall utility of the query functionality. This paper studies the direction-aware why-not spatial keyword top-k query problem. We propose efficient query refinement techniques to revive missing objects by minimally modifying users' direction-aware queries. Experimental studies demonstrate the efficiency and effectiveness of the proposed techniques.**

I. Introduction

The continued proliferation of location-based services (LBS) and the increasing number of web objects with both textual keywords and spatial location information combine to give prominence to spatial keyword queries [6], [7]. A *direction-aware spatial keyword top-k query* takes a user location, a set of keywords, and a search direction as arguments and retrieves the k objects in the search direction that are ranked highest according to a ranking function that considers both spatial proximity and textual similarity [7]. It is relevant to take into account the query direction in a number of scenarios. For example, a user walking to a supermarket may want to find an ATM in her walking direction. As another example, a user on a high-way may want to find a gas station or restaurant in the user's general driving direction.

There may be cases where users are not fully aware of the appropriate direction to feed to a spatial keyword top-k query; it may be difficult for a user to specify the direction that best captures her query intent. After a user issues a query and receives the result, the user may find the result is not as expected and that some desirable objects are unexpectedly missing. This may lead the user to question the overall result and to wonder whether other unknown relevant objects are also missing. It is thus relevant to provide explanations about desired but missing objects and to automatically suggest a refined query that includes the desired objects in its result.

*Example 1: After a busy day of sightseeing, Bob is tired and hungry. Walking back to the hotel, he issues a query to find the top-*3 *nearby "Sushi" restaurants. Surprisingly, he finds that the result contains only restaurants that are out of his way and that a restaurant on his way to the hotel that he visited yesterday is not in the result. Bob questions the overall result. Are the returned restaurants really the best, or do better options exist? Should the restaurants be searched in the general direction of his way to the hotel? How can he add a search direction so that the missing restaurant and possibly other good options appear in the result?*

This type of functionality relates to the query quality and is called *why-not* questions [1]. We adopt a *query refinement* model, which revises the original query so that missing objects can enter the result, to answer why-not questions on spatial keyword top-k queries. Existing works [3]–[5] adjust the preferences between spatial proximity and textual relevance [3] or suggest more accurate query keywords [4] to get the inclusion of missing objects in the query result. In this paper, we address the problem from a new perspective, *i.e.*, modifying the query direction as motivated by the above example.

We consider the problem in both the special case, where the initial query is a traditional spatial keyword query without a query direction, and the general case, where a query direction is initially specified. To achieve an efficient solution, we provide techniques to reduce the search space in both cases.

II. Preliminaries and Problem Formulation

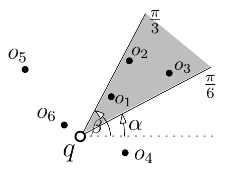
A. Spatial Keyword Top-k Queries

Let $\mathcal{D}$ denote a database of spatial objects. Each object $o \in \mathcal{D}$ is associated with a pair (loc, doc), where $o.loc$ is o's location and $o.doc$ is a set of keywords that describe the object.

A spatial keyword top-k query q takes four parameters $(loc, doc, \vec{w}, k)$. Here $q.loc$ is the query location, $q.doc$ is a set of query keywords, $q.k$ denotes the number of objects to retrieve, and $q.\vec{w} = \langle w_s, w_t \rangle$, where $0 \le w_s, w_t \le 1$ and $w_s + w_t = 1$, denotes the user's preferences between spatial proximity and textual relevance. The query retrieves the top-k objects from $\mathcal{D}$ ranked according to a scoring function that considers both spatial distance and textual similarity. We adopt a widely used ranking function [6]:

$$ST(o,q) = w_s \cdot (1 - SDist(o,q)) + w_t \cdot TSim(o,q), \quad (1)$$

where $SDist(o,q)$ and $TSim(o,q)$ are normalized spatial distance and textual similarity, respectively. The textual similarity $TSim(o,q)$ can be computed using an information retrieval model. Without loss of generality, we adopt the language model [3], [6] in this paper.



	$ST(o,q)$
$\underline{\mathbf{o_1}}$	0.72
o_2	0.3
$\mathbf{o_3}$	0.58
o_4	0.65
o_5	0.43
$\underline{o_6}$	0.83

(b)

Fig. 1. Top k and Direction-Aware Top-k Spatial Keyword Query

Given a query q, the rank of an object o is given in terms of Eqn. (1) as follows:

$$R(o,q) = |\{o' \in \mathcal{D} \mid ST(o',q) > ST(o,q)\}| + 1 \tag{2}$$

Definition 1: **Spatial Keyword Top-k Query.** A spatial keyword top-k query q returns a set $\mathcal{R}$ of k objects from $\mathcal{D}$, where $\forall o \in \mathcal{R}\ (\forall o' \in \mathcal{D} - \mathcal{R}\ (ST(o,q) \geq ST(o',q)))$.

In a *direction-aware spatial keyword top-k query* $q = (loc, doc, \vec{w}, k, d)$, an object can be a result only if it is located in a certain direction d of a query location. A direction is defined in terms of rays emanating from the query location [7]. We assume that the objects are mapped to a Cartesian coordinate system. We delineate a direction by the angles between two rays and the positive direction of the x-axis. The direction d in a query is a range (α, β).[1]

Definition 2: **Direction-Aware Spatial Keyword Top-k Query.** Let $\mathcal{D}_d$ denote the objects in $\mathcal{D}$ that are located in the angular region $d = (\alpha, \beta)$, and let $R(o,q,d)$ denote the rank of an object o under a query q. A direction-aware spatial keyword top-k query q returns a set $\mathcal{R}$ of k objects from $\mathcal{D}_d$, where $\forall o \in \mathcal{R}\ (\forall o' \in \mathcal{D}_d - \mathcal{R}\ (R(o,q,d) \leq R(o',q,d)))$.

Example 2: See Fig. 1 as an example, where (a) shows the locations of the query and objects, while (b) lists the ranking score of each object. Consider a top-2 query. A traditional query returns the objects with the highest scores among all the objects, i.e., o_6 and o_1. However, if the query has a direction, say, $(\frac{\pi}{6}, \frac{\pi}{3})$, the query considers only the objects in $(\frac{\pi}{6}, \frac{\pi}{3})$ and returns the top-2 objects among them, i.e., o_1 and o_3.

B. Direction-Aware Why-Not Spatial Keyword Query

Due to an improper setting of the query direction, after the user receives a query result, the user may find that one or more desired objects are unexpectedly missing. The user may then pose a follow-up *why-not* question with a set $M = \{m_1, m_2, ..., m_j\}$ of desired but missing objects, asking *why* these expected objects are missing and seeking a refined query $q' = \{loc, doc, \vec{w}, k', d'\}$ that includes the missing objects in its result. As simply modifying the direction in the initial query may not be able to revive the missing objects, the enlargement of k is also considered. We aim to provide the users with refined queries that minimally modify their initial queries. Specifically, the penalty of a refined query q' against the initial query q is defined as follows:

$$Penalty(q,q') = \lambda \cdot \frac{\Delta k}{\Delta k_{max}} + (1-\lambda) \cdot \frac{\Delta d}{\Delta d_{max}}, \tag{3}$$

[1]A direction (α, β) is the angular space passed through by rotation from α to β counterclockwise. We use open or closed intervals to denote whether a boundary direction is included. For the sake of presentation, we convert all directions to the space where $-\pi < \alpha \leq \pi$ and $\alpha \leq \beta \leq \alpha + 2\pi$. We say a direction $\theta \in (\alpha, \beta)$, if $\exists n \in \mathbb{Z}, \theta + 2n\pi \in (\alpha, \beta)$.

where $\lambda \in [0,1]$ is a user preference on modifying k versus d. Here, Δk_{max} and Δd_{max} denote the maximum possible modifications of k and d, respectively. They are used to normalize Δk and Δd. Since their settings would vary in different cases, we leave their definitions to the coverage of the corresponding cases in Sections III and III-C. We have $\Delta k = max(0, k' - k_0)$, since k' can remain as k_0 if a refined k' is smaller than k_0. As the query direction d is an angular space between the start angle α and the end angle β, we measure the modification from $d_0 = (\alpha_0, \beta_0)$ to $d' = (\alpha', \beta')$ in terms of how much d_0 is rotated and how much the size of d_0 is changed, *i.e.*, Δr and Δs. Formally, Δd is defined as follows:

$$\begin{aligned} \Delta d &= \gamma \cdot \Delta r + (1-\gamma) \cdot \Delta s \\ &= \gamma \cdot |\frac{\alpha' + \beta'}{2} - \frac{\alpha_0 + \beta_0}{2}| \\ &\quad + (1-\gamma) \cdot |(\beta' - \alpha') - (\beta_0 - \alpha_0)| \end{aligned} \tag{4}$$

The rotation of the direction Δr is measured as the difference between the angular bisectors, *i.e.*, $\frac{\alpha' + \beta'}{2}$ and $\frac{\alpha_0 + \beta_0}{2}$. Finally, $\gamma \in [0,1]$ is used to balance the changes to the rotation and the size of the direction.

Definition 3: **Direction-Aware Why-Not Spatial Keyword Top-k Query.** Given a set $\mathcal{D}$ of spatial objects, a missing object set M, an original direction-aware spatial keyword query $q = (loc, doc, \vec{w}, k_0, d_0)$, the *direction-aware why-not spatial keyword top-k query* returns the refined query $q' = (loc, doc, \vec{w}, k', d')$, with the lowest penalty according to Eqn. (3) and the result of which includes all objects in M.

C. Baseline Algorithm

Lemma 1: Given an initial query q and a set M of missing objects, a pair of a modified direction and a refined result cardinality (d', k') can be a candidate for the best refined query if and only if *(i)* $\forall m_i \in M\ (\theta_{m_i} \in d')$, where θ_{m_i} denotes m_i's angle; and *(ii)* $k' = R(M,q,d')$; or $R(M,q,d') \leq k' \leq k_0$, where $R(M,q,d') = max_{m_i \in M} R(m_i, q, d')$.

According to Lemma 1, given a refined direction d', we can always set $k' = R(M,q,d')$ to achieve the minimum penalty. This observation inspires a baseline solution as follows: *(i)* enumerate possible refined directions; *(ii)* for each candidate direction, process a direction-aware spatial keyword top-k query to determine the ranks of the missing objects; *(iii)* compute the penalty of each candidate direction and return the one with the minimum penalty.

III. Why-Not Solution: A Special Case

In this section, we assume a single missing object and study the why-not problem for the special case, where the initial query is a traditional query with no query direction.

A. Case Analysis

The traditional spatial keyword top-k query can be treated as a query with a direction in $\{d \mid d = [\alpha, \alpha + 2\pi), -\pi < \alpha \leq \pi\}$. The bisector of such a direction can be any ray around the query location. Thus, the modification in rotating the initial direction can be treated as 0, and Δd can be rewritten as follows:

$$\Delta d = 2\pi - (\beta' - \alpha') \tag{5}$$

Algorithm 1 Answering Why-Not Questions: Special Case

```
INPUT: Original query $q = (loc, doc, \vec{w}, k_0, d_0 = [-\pi, \pi))$,
       Missing object $m$
OUTPUT: Best refined query $q' = (loc, doc, \vec{w}, k', d')$
 1: compute $R(m, q, d_0)$ and record $m$'s dominators in set $S$
 2: $\theta_m \leftarrow$ **CalDirection**$(q, m)$ //calculate the angle of an object *w.r.t. q.loc*
 3: for each $o_i \in S$
 4:     $\theta[i] \leftarrow$ **CalDirection**$(q, o_i)$
 5: sort $\theta[i]$ in clockwise order w.r.t. $\theta_m$
 6: $d' \leftarrow d_0$, $k' \leftarrow R(m, q, d_0)$, $p_c \leftarrow \lambda$
 7: for $r = k_0$ to $R(m, q, d_0) - 1$ do
 8:     $\Delta k \leftarrow r - k_0$
 9:     if $\lambda \cdot \frac{\Delta k}{R(m,q,d_0)-k_0} \geq p_c$ then
10:         return $q' = (loc, doc, \vec{w}, k', d')$
11:     for $i = 1$ to $r$ do
12:         $\alpha' \leftarrow \theta[i]$
13:         $\beta' \leftarrow \alpha' + ((\theta[(R(m,q,d_0)-1)-(r-i)] - \theta[i] + 2\pi)\%2\pi)$
14:         compute the penalty $p$ for the current candidate direction according
            to Eqn. (6)
15:         if $p < p_c$ then
16:             $k' \leftarrow r$, $d' \leftarrow (\alpha', \beta')$, $p_c \leftarrow p$
17: return $q' \leftarrow (loc, doc, \vec{w}, k', d')$
```

Thus, the maximum possible modification of the direction is $\Delta d_{max} = 2\pi$. Δk_{max} can be estimated as $R(m, q, d_0) - k_0$, where $R(m, q, d_0)$ denotes the rank of the missing object m under the initial query, as a very naive method to revive the missing object is to increase k_0 until m is included in the result without adjusting the direction. As such, the penalty function becomes:

$$Penalty(q, q') = \lambda \cdot \frac{\Delta k}{R(m, q, d_0) - k_0} + (1 - \lambda) \cdot \frac{2\pi - (\beta' - \alpha')}{2\pi} \quad (6)$$

For a given k', the largest direction that could rank the missing object within top-k' is preferred.

B. Ranking Updates and Direction Modification

After the initial query is issued, the ranking score of each object is a constant. The rank of the missing object in the refined query is determined by the number of objects that scores better than m in the refined direction. Given a start angle α' of a refined direction, the following theorem holds.

Theorem 1: Consider an initial query q, a missing object m, and a refined start angle α'. Let θ_m denote the angle of m w.r.t. q. If a direction (α', β') is a candidate for the best refined query, it holds that: *(i)* $\theta_m \in (\alpha', \beta')$; and *(ii)* there exists an object o with angle β' such that $ST(o, q) > ST(m, q)$.

Similarly, for a given refined end angle β', the refined start angle α' should be set to $\beta' - 2\pi$, or should be the angle of an object that ranks higher than the missing object under the initial query.

Proposition 1: A refined direction (α', β') can result in the best refined query if: *(i)* $\theta_m \in (\alpha', \beta')$; and *(ii)* both the start angle α' and the end angle β' have an object that ranks higher than the missing object under the initial query.

C. Answering Why-Not

Proposition 1 limits the candidate directions to a finite set and thus enables enumeration. The pseudo-code of the algorithm for answering the why-not questions in the special case is given in Algorithm 1.

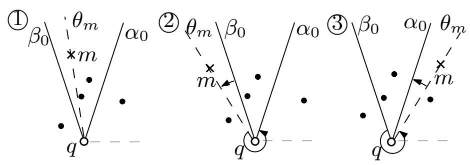


Fig. 2. Cases of the reason about m's missing

IV. Why-Not Solution: General Case

We proceed to analyze the general case, where initial queries are specified with direction requirements.

A. Case Analysis

Unlike the special case, here the bisector of the initial direction is fixed, *i.e.*, $\frac{\alpha_0+\beta_0}{2}$. According to Eqn. (4), the maximum possible modification of the direction is as follows:

$$\Delta d_{max} = \gamma \cdot \pi + (1 - \gamma) \cdot Max\{\beta_0 - \alpha_0, 2\pi - (\beta_0 - \alpha_0)\} \quad (7)$$

The reason why an expected object m is missing from the result of an initial query $q = (loc, doc, \vec{w}, k_0, d_0)$ can be discussed in two scenarios: *(i)* m is in q's query direction d_0, but has a worse rank than k_0; *(ii)* m is outside q's direction d_0. Fig. 2 illustrates an expected object m missing from a query result in different cases. Case ① belongs to scenario *(i)*. We further divide scenario *(ii)* into two sub-cases, *i.e.*, ② and ③. In ②, θ_m is closer to β_0, since $(\theta_m - \beta_0 + 2\pi)\%2\pi$ is smaller than $(\alpha_0 - \theta_m + 2\pi)\%2\pi$. In ③, θ_m is closer to α_0.

1) Case ①: Here the missing object is in the query direction. The missing object m does have a rank $R(m, q, d_0)$ under the initial query. The maximum possible modification to k is $\Delta k_{max} = R(m, q, d_0) - k_0$. As such, the penalty function for the why-not problem in Case ① is the following:

$$\begin{aligned} Penalty(q, q') &= \lambda \cdot \frac{\Delta k}{R(m, q, d_0) - k_0} \\ &+ (1 - \lambda) \cdot \frac{\Delta d}{\gamma \cdot \pi + (1 - \gamma) \cdot Max\{\beta_0 - \alpha_0, 2\pi - (\beta_0 - \alpha_0)\}} \end{aligned} \quad (8)$$

Theorem 2: Consider an initial query q with direction $d_0 = (\alpha_0, \beta_0)$ and a missing object m in d_0. Let θ_m be the angle of m *w.r.t.* q. A direction (α', β') can result in the best refined query if: *(i)* $\theta_m \in (\alpha', \beta')$; and *(ii)* $(\alpha', \beta') \subset [\theta_m - (\beta_0 - \alpha_0), \theta_m + (\beta_0 - \alpha_0)]$.

Theorem 2 reduces the search space for the refined directions to a smaller candidate space, making the processing of the why-not query more efficient. We denote this **C**andidate **S**pace as $CS(d')$, *i.e.*, $CS(d') = [\theta_m - (\beta_0 - \alpha_0), \theta_m + (\beta_0 - \alpha_0)]$.

2) Case ②: Similar to Case ①, the search space for Case ② can be reduced by Theorem 3 to a smaller candidate space $CS(d') = [\alpha_0 - (\theta_m - \beta_0), \theta_m + (\beta_0 - \alpha_0)]$.

Theorem 3: Consider an initial query q with a direction $d_0 = (\alpha_0, \beta_0)$ and a missing object m outside d_0. Let θ_m denote the angle of m w.r.t. q. If $(\theta_m - \beta_0 + 2\pi)\%2\pi < (\alpha_0 - \theta_m + 2\pi)\%2\pi$, a direction (α', β') can result in the best refined query if: *(i)* $\theta_m \in (\alpha', \beta')$; and *(ii)* $(\alpha', \beta') \subset [\alpha_0 - (\theta_m - \beta_0), \theta_m + (\beta_0 - \alpha_0)]$.

In this case, the expected object is outside the initial direction. It is thus impossible to revive the missing object by simply enlarging k. Nevertheless, Theorem 3 implies that the refined direction belongs to $CS(d')$. We set the maximum possible refined k' to be the rank of m in $CS(d')$, denoted by $R(m,q,CS(d'))$. As this rank could be smaller than k_0, to normalize Δk and avoid the denominator Δk_{max} being 0, the maximum modification on k is given as follows:

$$\Delta k_{max} = max\{R(m,q,CS(d')) - k_0, 1\}$$

Thus, the penalty function for the why-not problem in Case ② is represented as follows:

$$\begin{aligned} Penalty(q,q') &= \lambda \cdot \frac{\Delta k}{max\{R(m,q,CS(d')) - k_0, 1\}} \\ &+ (1-\lambda) \cdot \frac{\Delta d}{\gamma \cdot \pi + (1-\gamma) \cdot Max\{\beta_0 - \alpha_0, 2\pi - (\beta_0 - \alpha_0)\}}. \end{aligned} \tag{9}$$

3) Case ③: Case ③ is symmetric to Case ② except that the candidate space for the refined direction in this case is $CS(d') = [\theta_m - (\beta_0 - \alpha_0), \beta_0 + (\alpha_0 - \theta_m)]$.

B. Answering Why-Not

The basic idea of the algorithm for the general case is similar to that of the special case. The main difference is that, a candidate refined direction $d' = [\alpha', \beta']$ that ranks the missing object at a given k' belongs to several range pairs of α' and β'. To compute the optimal refined query for a given k', we solve a linear programming problem for each range pair of α' and β', with the objective function set as the corresponding penalty function:

$$\begin{aligned} min \quad \Delta d \quad &= \gamma \cdot |\tfrac{\alpha'+\beta'}{2} - \tfrac{\alpha_0+\beta_0}{2}| \\ &+(1-\gamma) \cdot |(\beta' - \alpha') - (\beta_0 - \alpha_0)| \\ s.t. \quad &\begin{cases} \alpha' \in (\theta[i], \theta[i-1]] \\ \beta' \in [\theta[i-1] + (\theta[|S|-(k'-i)+1] - \theta[i-1] + 2\pi)\%2\pi, \\ \qquad \theta[i-1] + (\theta[|S|-(k'-i)] - \theta[i-1] + 2\pi)\%2\pi), \end{cases} \end{aligned}$$

where $i \in \mathbb{Z}$ and $1 \le i \le min\{k', t+1\}$. Note that the β' range is computed in a way that ensures that it satisfies the constraint $\beta' \in [\alpha', \alpha' + 2\pi)$.

V. Empirical Study

A. Experimental Setup

The experiments run on a PC with an Intel Core i5 2.7GHz CPU and 8GB memory running Windows 7 OS. The algorithms are implemented in Java. We extend an existing algorithm [6] to process direction-aware spatial keyword top-k queries by examining whether each accessed MBR or object is in the query direction. The index structure adopted, *i.e.*, the IR-tree, is disk-resident. The page size is set to 4KB and the fanout is 100. We report the query time of 1,000 randomly generated queries.

We use two real datasets, EURO and GN. EURO is a dataset of points of interest in Europe (www.allstays.com); and GN is obtained from the US Board on Geographic Names. Table I gives more details about the datasets.

TABLE I. Dataset Information

Dataset	# objects	# distinct words	Avg. # words per object
EURO	162,033	35,315	18
GN	1,868,821	222,407	4

The initial query is a top-10 query with 4 keywords. The missing object is the object that ranks at 101 under the initial query without taking into account a query direction. The default values of $\vec{w}$ and γ are $<0.5, 0.5>$ and 0.5, respectively.

B. Experimental Results

1) Baseline vs. Algorithm 1: We first compare Algorithm 1 against the baseline method presented in Section II-C for the special case. The initial query is without a query direction. Fig. 3 shows the runtime, where *PMK*, *PMD* and *NM* stand for "Prefer Modifying K ($\lambda = 0.1$)", "Prefer Modifying Direction ($\lambda = 0.9$)" and "Never Mind ($\lambda = 0.5$)". Algorithm 1 outperforms the baseline method by two orders of magnitude. The reason is that the baseline method needs to invoke a spatial keyword top-k query to determine the missing object's rank for each candidate direction. In contrast, Algorithm 1 is able to calculate the penalty of a candidate direction in constant time.

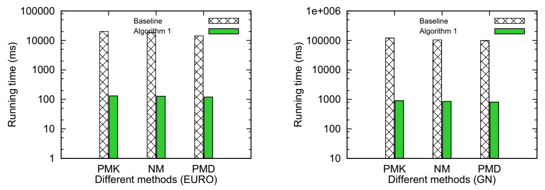


Fig. 3. Running time vs. Baseline & Algorithm 1

2) Varying d_0: To examine the performance of the algorithm proposed for the general case, we test queries with different initial direction sizes, from $\frac{\pi}{6}$ to 2π. Fig. 4 plots the query time. A larger d_0 results in a larger candidate space $CS(d')$ for the refined query directions. Computing the rank of the missing object in a larger candidate space consumes more time. This explains why the runtime increases with the size of d_0. Nevertheless, the algorithm scales well as d_0 increases. For example, the running time only increases 20% when the direction varies from $\frac{\pi}{6}$ to 2π. Interested readers are referred to [2] for more detailed experimental results.

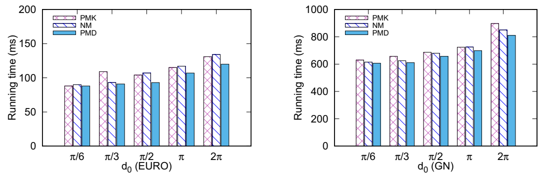


Fig. 4. Running time vs. d_0

Acknowledgements. This work is supported by HK-RGC Grants 12201615 & 12244916 and NSFC Grant 61602420.

References

[1] A. Chapman and H. V. Jagadish. Why not?. In *SIGMOD*, 2009.

[2] L. Chen, Y. Li, J. Xu, and C. S. Jensen. Direction-aware why not spatial keyword top-k queries. *HKBU Technical Report*, 2017.

[3] L. Chen, X. Lin, H. Hu, C. S. Jensen, and J. Xu. Answering why-not questions on spatial keyword top-k queries. In *ICDE*, 2015.

[4] L. Chen, J. Xu, X. Lin, C. S. Jensen, and H. Hu. Answering why-not spatial keyword top-k queries via keyword adaptation. In *ICDE*, 2016.

[5] L. Chen, J. Xu, C. S. Jensen, and Y. Li. YASK: A why-not question answering engine for spatial keyword query services. In *PVLDB*, 2016.

[6] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. In *PVLDB*, 2009.

[7] G. Li, J. Feng, and J. Xu. DESKS: Direction-aware spatial keyword search. In *ICDE*, 2012.