2014 IEEE 15th International Conference on Mobile Data Management

Social-Aware Top-k Spatial Keyword Search

Dingming Wu, Yafei Li, Byron Choi, and Jianliang Xu
Department of Computer Science
Hong Kong Baptist University
Email: {dmwu,yfli,bchoi,xujl}@comp.hkbu.edu.hk

***Abstract*—The boom of the spatial web has enabled spatial keyword queries that take a user location and multiple search keywords as arguments and return the objects that are spatially and textually relevant to these arguments. Recently, utilizing social data to improve search results, normally by giving a higher rank to the content generated or consumed by the searcher's friends in the social network, has been studied in the information retrieval (IR) community. However, little attention has been drawn to the integration of social factors into spatial keyword query processing. In this paper, we propose a novel spatial keyword query, Social-aware top-k Spatial Keyword (SkSK) query, which enriches the semantics of the conventional spatial keyword query by introducing a new *social relevance* attribute. A hybrid index structure, called Social Network-aware IR-tree (SNIR-tree), is proposed for the processing of SkSK queries. To further improve the query response time, an x-hop localized algorithm is developed. Empirical results demonstrate that the proposed index and algorithms are capable of excellent performance.**

I. Introduction

In recent years, the web has been accessed increasingly by mobile users due to a proliferation of Internet-worked mobile devices such as smartphones. Meanwhile, positioning technologies have been increasingly available for mobile devices, e.g., GPS receivers and Wi-Fi. These developments enable a spatial web where contents and users are associated with geographical locations, resulting in a wide range of location-based services, e.g., map services, local search, and local advertisements.

In the spatial web, a spatial keyword query takes a user location and multiple search keywords as arguments and returns the objects that are spatially and textually relevant to these arguments. Several proposals in the literature have studied some variants of the spatial keyword query, considering different semantics of the spatial or textual relevance between queries and objects. Some work [11], [14] treat the textual relevance as Boolean predicates, selecting the objects that contain the query keywords and satisfy the spatial constraint in the query. Some work [12], [19], [22] integrate the spatial and textual relevance into a ranking function.

However, ranking only according to the relevance between the query arguments and objects adopted by existing work may not satisfy different users' information needs. As an example, searching for "mouse" in "New Mexico" by a biologist (getting some information about human plague in New Mexico) has a completely different meaning from searching by a programmer who is interested in computer peripherals (buying a mouse in New Mexico). With the popularity of location-based social networking services, such as Twitter, Foursquare and Facebook, the social networks can be utilized to help improving the quality of search results that might subjectively satisfy the searcher's needs. It is motivated by the fact that we usually turn to our friends for recommendations of books, movies, or restaurants, since closely related people have similar interests. Ye et al. [29] show that social influence is beneficial for item recommendation. The idea behind is that a user's friends may share common interests with the user, and have influence on the user's decisions.

Utilizing social data to improve search results, normally by giving a higher rank to the content generated or consumed by the searcher's friends in the social network, has been studied in the information retrieval (IR) community [1], [3]. This type of search not only has applications such as name, entity, or content search on social networks [23], and social question and answering [17], but also is very effective for personalization of web search [8]. Recently, Facebook announced "Graph Search", a way to search all of Facebook's content for queries tailored to users' profiles, e.g., searching through which dentists your friends used for. However, no emphasis has been placed explicitly on users' social influence in spatial keyword search.

This paper aims to integrate social data into spatial keyword search, by introducing a social relevance for each object with regard to the query user. Thus, given a query q, the ranking score of an object p is determined by a function $f(\|q\ p\|, tr_q(p), sd_q(p))$ considering not only the spatial and textual relevance ($\|q\ \ p\|$ and $tr_q(p)$) but also the social relevance ($sd_q(p)$). Within a social network context, each object has a set of fans who have positive attitudes towards the object. The social relevance of an object is determined by the relationship between its fans and the query user in the social network. Figure 1 illustrates how social relevance affect the ranking of objects. There are two query users u_q and u'_q issuing the same query keywords at the same query location, shown in Figure 1(b). Hence, the spatial and textual relevance of each of the three objects p_1, p_2, and p_3 are the same for u_q and u'_q, respectively. In other words, the rankings of the three objects with regard to u_q and u'_q depend on the social relevance of the objects. Figure 1(a) shows the social network of the two users. The users following each object are the fans of the object. Let the three objects p_1, p_2, and p_3 have the same textual relevance. Traditional spatial keyword search technique ranks p_2 as the top-1 result. If considering the social relevance, object p_1 may be returned as the top-1 result to user u_q, while the top-1 result for u'_q may be p_3.

To the best of our knowledge, we are the first to study social-aware spatial keyword search. We propose a new type of query, called the *Social-aware top-k Spatial Keyword (SkSK) query* that retrieves a list of k objects ranked according to their spatial, textual, and social relevance. To process the SkSK

978-1-4799-5705-7/14 $31.00 © 2014 IEEE
DOI 10.1109/MDM.2014.35

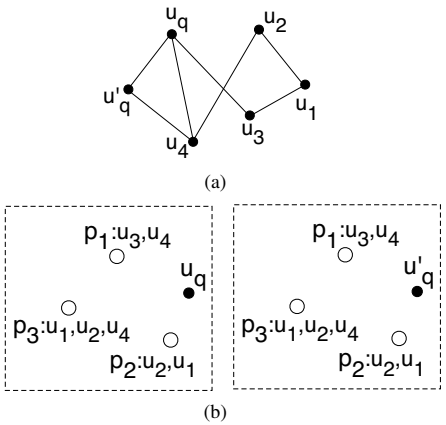


Fig. 1. Effect of Social relevance

query, a simple approach is to use a spatial keyword search technique to generate a number of top candidate objects based on the spatial and textual relevance and then compute the social relevance of the candidate objects. However, this approach is not efficient since it is not easy to determine the number of candidate objects needed from the first step in order to ensure that top-k results are found in the end.

In this paper, we propose a hybrid index structure, called Social Network-aware IR-tree (SNIR-tree) that integrates locations, text documents, and social information. Each entry in each node of the SNIR-tree records a summary of the location information, the textual content, and the social information of all the objects in the sub-tree rooted at the entry. With the SNIR-tree, a threshold-based algorithm is developed to efficiently process SkSK queries. The query processing algorithm is able to estimate the spatial distance, the text relevance scores, and the social relevance scores of a query to the objects in any sub-tree. It is able to simultaneously prune the search space according to the three measures, i.e., spatial proximity, textual relevance, and social relevance. To further improve the performance of processing SkSK queries on the SNIR-tree, an x-hop localized algorithm is proposed that ignores the fans x-hop away from the query user when computing the social relevance of objects. It is able to reduce the effect of the sparsity of the fan-object space and derives tight bounds on the social relevance in the SNIR-tree.

Our contributions made in this paper are summarized as follows:

- We propose a new SkSK query to integrate the social influence into spatial keyword search.
- A hybrid index SNIR-tree and a threshold-based algorithm are developed.
- An x-hop localized algorithm is proposed to further improve the performance of query processing, while producing similar results to the threshold-based algorithm.
- Extensive empirical experiments are conducted to evaluate the performance of our proposals.

The rest of this paper is organized as follows. The problem setting and definition are presented in Section II. Then, we propose the hybrid index structure SNIR-tree and an algorithm for the processing of SkSK queries in Section III and an x-hop localized algorithm in Section IV. Next, we study the performance of our proposals on real datasets in Section V, followed by related work in Section VI and conclusions in Section VII.

II. Problem Definition

We consider a data set $\mathcal{D}$ in which each object $p \in \mathcal{D}$ is a triple $\langle \lambda, \psi, \digamma \rangle$ of a point location $p.\lambda$, a text description or document $p.\psi$ (e.g., the facilities and menu of a restaurant), and a set of fans $p.\digamma$, where a fan $u \in p.\digamma$ is a user who has a positive attitude towards object p (e.g., liking, recommending or sharing). A Social-aware top-k Spatial Keyword (SkSK) query $q = \langle \lambda, \psi, k, S \rangle$ takes four arguments: a point location λ, a set of keywords ψ, a number of requested objects k, and the social network S of the user who issues the query. A social network S is modeled as an undirected graph, where each node represents a user and each edge indicates the relationship, connection, or interaction between two users. We define the social distance between two users $\|u_1\ u_2\|_s$ as the length of the shortest path $p(u_1, u_2)$ between u_1 and u_2 in the social network. Figure 2 illustrates an example social network, where the social distance between u_1 and u_2 is 2.

An SkSK query returns a list of k objects from $\mathcal{D}$ that minimize the ranking value (according to Equation 1) and that are in ascending order of their ranking values:

$$rank_q(p) = \frac{\|q\ p\|}{tr_q(p) \cdot sd_q(p)}, \tag{1}$$

where $\|q\ p\|$ is the Euclidean distance between query q and object p and $tr_q(p)$ is the text relevance of object p given the keywords in query q, which can be computed by any monotone information retrieval model, e.g., language models [21]. Function $sd_q(p)$ computes the social relevance of p with regard to q that favors the object having more fans close to the query user in the social network, defined as

$$sd_q(p) = 1 + \sum_{u \in p.\digamma} \alpha^{\|u_q\ u\|_s}, \tag{2}$$

where $\alpha \in [0, 1)$ is a damping factor. Constant 1 guarantees that the social relevance never equals 0. The exponential decay component $\alpha^{\|u_q\ u\|_s}$ is also used in other applications, such as voting in social networks [4] and PageRank [2]. Note that this paper's proposal is not limited to the ranking function introduced above. It is applicable to any monotone-form ranking function.

Example 2.1: Figure 3(a) shows the locations of a set of objects $\mathcal{D} = \{p_1, p_2, p_3, p_4\}$. Let the query q shown in Figure 3(a) be $q.\psi = \{a, b\}$, $q.k = 2$. Let the query user be u_1 and its social network $q.S$ as shown in Figure 2. The number in brackets next to each object is its text relevance to the query keywords $q.\psi$ that are computed on-the-fly using the text relevance function $tr_q(p)$. Figure 3(b) shows the Euclidean distances and the fans of each object. Let the damping factor α be fixed at 0.5. The social relevance of p_1 w.r.t. q is computed as $1 + 0.5^2 + 0.5^2 = 1.5$. Similarly, we have $sd_q(p_2) = 2.625$, $sd_q(p_3) = 1.0625$, and $sd_q(p_4) = 1.75$. The result of query

q is $\langle p_2, p_4 \rangle$ according to function $rank_q(\cdot)$ (Equation 1). The ranking values of p_2 and p_4 are 0.18 ($= 0.11/0.23/2.625$) and 0.32 ($= 0.14/0.25/1.75$), respectively.

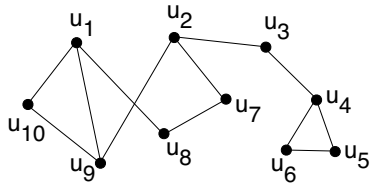
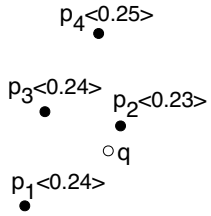


Fig. 2. Example Social Network



(a)

	$\|q\ p_i\|$	$p_i.F$
p_1	0.14	u_2, u_7
p_2	0.11	u_1, u_3, u_9
p_3	0.13	u_5, u_6
p_4	0.14	u_7, u_{10}

(b)

Fig. 3. Example SkSK Query

Problem Statement: We tackle the problem of efficiently answering SkSK queries, i.e., given a query q, we retrieve a ranked list of k objects according to their ranking scores as computed by the ranking function $rank_q(p)$ (Equation 1).

III. SNIR-Tree based Approach

Recently, several approaches have been developed for the processing of spatial keyword queries that consider the Euclidean distances and the text relevance of objects, e.g., the IR-tree and its variants [12], [19], [26] and S2I [22]. Additionally, the shortest path queries [9], [24] have been studied extensively in the literature. An SkSK query considers the Euclidean distances, the text relevance, and the social relevance of objects, where the social relevance of an object depends on the lengths of the shortest paths between the fans of the object and the query user. This is the first work that leverages both techniques for the efficient processing of SkSK queries.

A. SNIR-Tree

For the sake of efficiency, we propose a hybrid indexing structure, the Social Network aware IR-tree (SNIR-tree), that supports the simultaneous computation of the Euclidean distance, the text relevance, and the social relevance. It is a tree based structure which is able to prune the search space using the score bounds of non-leaf nodes. The SNIR-tree is extended from an IR-tree [12], each entry in each node of which is enriched with a set of fans that is the union of the fans of the objects contained in the subtree rooted at the entry.

In the SNIR-tree, each leaf node contains a number of entries of the form $\langle ptr, \Lambda, \psi, F \rangle$, where ptr refers to an object p in data set $\mathcal{D}$, Λ is the Minimum Bounding Rectangle (MBR) of object p, ψ is the identifier of the document of object p, and F refers to a set of fans of object p. Each leaf node also contains a pointer to an inverted file for the text documents of the objects stored in the node. An inverted file index has two main components:

1) A vocabulary of all distinct words appearing in the document of some object.
2) A posting list for each word t, i.e., a sequence of the identifiers of the objects whose documents contain t.

Each non-leaf node N contains a number of entries of the form $\langle ptr, \Lambda, \psi, F \rangle$ where ptr is a reference to a child node of N, Λ is the MBR of all rectangles in entries of the child node, ψ is a reference to a pseudo document that represents all documents in the entries of the child node, which enables derivation of an upper bound on the text relevance to a query of any object contained in the subtree rooted at this entry, and F refers to a set of fans that is the union of the fans of entries of the child node. Each non-leaf node contains a pointer to an inverted file on the pseudo documents of the entries stored in the node.

Example 3.1: Figure 4(a) illustrates an SNIR-tree on 9 spatial objects. Figure 4(b) shows the contents of the inverted files associated with the nodes. As a specific example, the weight of the term c in entry R_2 of node R_5 is 7, which is the maximal weight of the term in the three documents in node R_2. For ease of understanding, we use term frequency to represent the weight of word t in the running example of the paper. How the fans of entries are computed and stored in an SNIR-tree is illustrated in Figure 4(a). As a specific example, the fan sets of object p_1 and p_2 are $\{u_1, u_3, u_5\}$ and $\{u_3, u_5\}$, respectively. Then the fan set of their parent R_1 is the union of the fan sets of p_1 and p_2, i.e., $\{u_1, u_3, u_5\}$.

The SNIR-tree inherits an important feature from the IR-tree, i.e., the text relevance of a non-leaf entry is an upper bound on the text relevance to a query of any object contained in the subtree rooted at the entry, to be presented in Theorem 1.

Theorem 1: **[Monotonicity, text relevance function]** ([12]) Given a query q and a non-leaf entry e with its rectangle $e.\Lambda$, we have $\forall p \in e.\Lambda\ (tr_q(e) \geq tr_q(p))$.

As an example in Figure 4, given any query q, $tr_q(R_5) \geq tr_q(R_1) \geq tr_q(p_1)$. The construction of the fan set of a non-leaf entry allows derivation of an upper bound on the social relevance of any object contained in the subtree rooted at the entry, which is guaranteed by the following theorem:

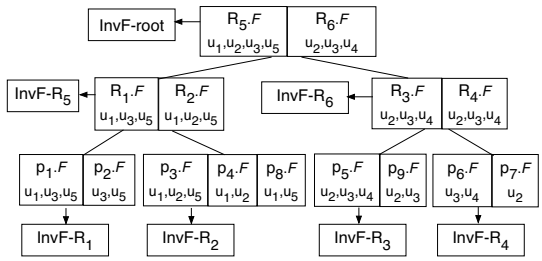
Theorem 2: Given a query q and a non-leaf entry e whose child node encloses m entries $CE = \{ce_i, 1 \leq i \leq m\}$, the following is true: $\forall ce_i \in CE(sd_q(ce_i) \leq sd_q(e))$.

Proof: Since ce_i is in the child node of e, the fan set of ce_i is a subset of the fan set of e, i.e., $ce_i.F \subseteq e.F$. According to the definition of the social relevance $sd_q(\cdot)$, we have

$$\begin{aligned} sd_q(ce_i) &= 1 + \sum_{u \in ce_i.F} \alpha^{\|u_q\ u\|_s} \\ &\leq 1 + \sum_{u \in e.F} \alpha^{\|u_q\ u\|_s} \\ &= sd_q(e), \end{aligned}$$

and thus complete the proof. ∎

A salient feature of the SNIR-tree is that it has a nice property (Theorem 3) that the ranking score of an entry at



(a) Structure of an SNIR-Tree

InvF-root	*InvF-R_5*	*InvF-R_6*
a: $(R_5, 7), (R_6, 4)$	a: $(R_1, 5), (R_2, 7)$	a: $(R_3, 4), (R_4, 1)$
b: $(R_5, 5), (R_6, 4)$	b: $(R_1, 5), (R_2, 3)$	b: $(R_4, 4)$
c: $(R_5, 7), (R_6, 4)$	c: $(R_1, 5), (R_2, 7)$	c: $(R_3, 4), (R_4, 4)$
d: $(R_5, 1), (R_6, 1)$	d: $(R_2, 1)$	d: $(R_4, 1)$

InvF-R_1	*InvF-R_2*	*InvF-R_3*	*InvF-R_4*
a: $(p_1, 5)$	a: $(p_3, 7)$	a: $(p_5, 4), (p_9, 3)$	a: $(p_7, 1)$
b: $(p_2, 5)$	b: $(p_8, 3)$		b: $(p_6, 4), (p_7, 1)$
c: $(p_1, 5), (p_2, 5)$	c: $(p_4, 7), (p_8, 3)$	c: $(p_5, 4), (p_9, 3)$	c: $(p_6, 3), (p_7, 4)$
	d: $(p_3, 1), (p_4, 1)$		d: $(p_7, 1)$

(b) Content of the Inverted Files of the Nodes

Fig. 4. Example of an SNIR-Tree

higher level in the SNIR-tree is a lower bound on the ranking scores of the entries in its subtree. These lower bounds are used to direct the search as shown in Section III-B.

Theorem 3: Given a query q and a non-leaf entry e whose child node encloses m entries $CE = \{ce_i, 1 \leq i \leq m\}$, the following is true: $\forall ce_i \in CE(rank_q(ce_i) \geq rank_q(e))$.

Proof: Since ce_i is in the child node of e, i.e., being enclosed in the rectangle of e, the minimum Euclidian distance between q and e is no larger than the minimum Euclidian distance between q and ce_i, i.e., $\|q\ e\| \leq \|q\ ce_i\|$. According to Theorems 1 and 2, we have $tr_q(ce_i) \leq tr_q(e)$ and $sd_q(ce_i) \leq sd_q(e)$. Hence, we have

$$\begin{aligned} rank_q(ce_i) &= \frac{\|q\ ce_i\|}{tr_q(ce_i) sd_q(ce_i)} \\ &\geq \frac{\|q\ e\|}{tr_q(e) sd_q(e)} \\ &= rank_q(e). \end{aligned}$$

■

B. Query Processing

To process SkSK queries with the SNIR-tree, we exploit the best-first traversal algorithm (e.g., [16]) for retrieving the top-k objects. With the best-first traversal algorithm, a priority queue is used to keep track of the entries referring to nodes or objects that have yet to be visited. The ranking scores $rank_q(\cdot)$ of entries are used as the keys. When deciding which node or object to visit next, the algorithm picks the entry with the smallest ranking score in the set of all entries that have yet to be visited. The algorithm terminates when the top-k objects (ranked according to Equation 1) have been found. Algorithm 1 shows the pseudo-code. We adopt the Breadth First Search (BFS) to determine the social relevance of visited entries (line 11).

C. Update of the SNIR-Tree

Since the SNIR-tree is based on the IR-tree, the update, including insertion and deletion of an object, is exactly as in the IR-tree. In addition, the fans of each entry in the SNIR-tree can be updated easily.

Algorithm 1 SNIRQP(Query q, Integer k, SNIR-tree $index$)

```
1:  Queue ← NewPriorityQueue();
2:  Queue.Enqueue(index.RootNode, 0);
3:  while not Queue.IsEmpty() do
4:      Entry e ← Queue.Dequeue();
5:      if e refers to an object then
6:          Add e to the top-k result;
7:          if k objects have been found then
8:              return the top-k result;
9:      else                                  ▷ e refers to a node
10:         for each entry e' in node e do
11:             Compute its social relevance sd_q(e') using BFS;
12:             Queue.Enqueue(e', rank_q(e'));
```

IV. x-Hop Localized Algorithm

In the SNIR-tree, we observe that opportunities for the performance improvement exist in the following two cases.

Case I: late stop.
Given a query, if the Euclidean distances and the text relevance of the objects in some leaf nodes are similar, their social relevance play important roles in their ranking scores. The social relevance (ranking scores) of the entries in their parent node are larger (smaller) than the social relevance (ranking scores) of those objects. Algorithm 1 will not stop even if the top-k objects have been encountered. They will be inserted into the priority queue, since some non-leaf entries have smaller ranking scores. The algorithm terminates when all these non-leaf entries in the front of the priority queue are processed, i.e., their children nodes are loaded, so that high I/O cost and computation cost are incurred. We use Example 4.1 to illustrate this case.

Example 4.1: Suppose the query user is u_{10} in the social network shown in Figure 2, retrieving the top-1 result. Let α in Equation 2 be 0.5. Figure 5 shows a part of an example SNIR-tree. To simplify the example, we assume the text relevance of each entry in the SNIR-tree is 1. The numbers in the parentheses are the Euclidean distances from the entries to the query user. The social relevance of entries R_1, R_2, and R_3 are 1.66 ($= 1+0.5^1+0.5^3+0.5^5$), 1.78 ($= 1+0.5^1+0.5^2+0.5^5$), and 1.69 ($= 1 + 0.5^1 + 0.5^3 + 0.5^4$), respectively, based on their fans shown in Figure 5. According to Algorithm 1, entries R_1, R_2, and R_3 are inserted in the priority queue with their

ranking scores 0.06 ($= 0.1/1/1.66$), 0.08 ($= 0.15/1/1.78$), and 0.08 ($= 0.135/1/1.69$). Entry R_1 is firstly dequeued and object p_1 and p_2 in its child node are inserted into the priority queue with ranking scores 0.087 and 0.17. In this example, object p_1 is the top-1 result. However, the algorithm cannot stop, since the ranking scores of R_2 and R_3 in the priority queue are smaller than that of p_1. The children nodes of R_2 and R_3 have to be processed before reporting p_1 as the top-1 result.

Case II: long search path.
The social relevance of a non-leaf entry e can sometimes be a loose bound, much larger than the social relevance of the entries in its child, if the fan sets of the entries in the child node are quite different. When entry e is dequeued from the priority queue, we expect the result can be retrieved in its child node. However, the ranking scores of the entries in the child node are much worse than that of e. Then other entries in the priority queue with better ranking scores need to be processed. The search path is long, since the leaf node where the result located cannot be quickly found. Example 4.2 illustrates this case.

Example 4.2: Suppose the query user is u_{10} in the social network shown in Figure 2, retrieving the top-1 result. Let α in Equation 2 be 0.5. Figure 6 shows a part of an example SNIR-tree. To simplify the example, we assume the text relevance of each entry in the SNIR-tree is 1. The numbers in the parentheses are the Euclidean distances from the entries to the query user. The social relevance of entries R_1 and R_2 are 1.66 ($= 1 + 0.5^1 + 0.5^3 + 0.5^5$) and 1.97 ($= 1+0.5^1+0.5^2+0.5^3+0.5^4+0.5^5$), respectively, based on their fans shown in Figure 6. According to Algorithm 1, entries R_2 and R_1 are inserted in the priority queue with their ranking scores 0.056 ($= 0.11/1/1.97$) and 0.06 ($= 0.1/1/1.66$). Entry R_2 is firstly dequeued and object p_3, p_4 and p_5 in its child node are inserted into the priority queue. In this example, object p_1 is the top-1 result. However, the algorithm traverses another search path via R_2, rather than the shortest one via R_1, since the ranking score of R_2 is smaller than that of R_1. The children nodes of entries with smaller ranking scores than that of R_1 have to be processed before reporting p_1 as the top-1 result.

Based on these observations, we propose an algorithm that computes social relevance for each entry in the SNIR-tree based on Equation 3 that only considers fans within x hops. The intuition is that the friends far away (more than x hops) from the query user have little effect on the result. This x-hop localized algorithm retrieves similar results as Algorithm 1, but achieving good search performance.

Specifically, the x-hop localized algorithm differs from Algorithm 1 in the social relevance computation. Equation 3 is used to compute the social relevance of each entry in the SNIR-tree, where parameter x affects the result. The larger the x is, the more similar the result is to Algorithm 1. When x approaching ∞, the x-hop localized algorithm finds the same result as does Algorithm 1.

$$sd_q(e) = 1 + \sum_{u \in e.F \wedge \|u_q\ u\|_s \leq x} \alpha^{\|u_q\ u\|_s}. \quad (3)$$

A smaller x tempers or mitigates the effect of the fans far from the query user in the social relevance. It reduces the effect of the sparsity of the fan-object space. Only the fans close to the query user are considered. Hence, the social relevance of non-leaf entries become tight bounds on the social relevance of the entries in their children nodes.

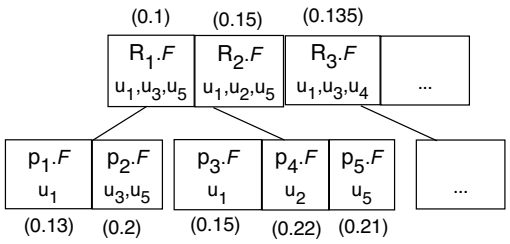


Fig. 5. Late Stop Example

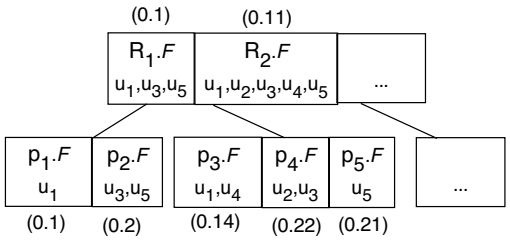


Fig. 6. Long Search Path Example

Applying the x-hop localized algorithm in Example 4.1, if we set x to 1, only u_1 is considered when computing the social relevance. Entries R_1, R_3, and R_2 are inserted in the priority queue with their ranking scores 0.07, 0.09, and 0.1. Entry R_1 is firstly dequeued and objects p_1 and p_2 in its child node are inserted into the priority queue with ranking scores 0.087 and 0.17. Then p_1 is reported as the top-1 result. The x-hop localized algorithm terminates earlier than Algorithm 1 does. Recall that Algorithm 1 processes the children nodes of entries R_2 and R_3 before reporting p_1. Similarly, if we set x to 1 in Example 4.2, entries R_1 and R_2 are inserted in the priority queue with their ranking scores 0.07 and 0.073. Entry R_1 is firstly dequeued and object p_1 is reported as the top-1 result. The x-hop localized algorithm quickly finds the node containing the results when compared to Algorithm 1. Recall that Algorithm 1 visits the child node of R_2 before finding the top-1 result.

In the experiment evaluation, we study how different values of x affect the search performance and measuring the similarity of the query results produced by the x-hop localized algorithm and Algorithm 1. In the above two examples, the x-hop localized algorithm retrieves the same result as Algorithm 1 does. According to our empirical study, the x-hop localized algorithm achieves much better performance. And the results are slightly different from Algorithm 1.

V. Empirical Study

We conduct empirical studies to evaluate our proposals. Section V-A presents datasets, queries, parameters, and the platform used in the experiments. The proposals for SkSK queries are evaluated in Section V-B.

A. Experimental Setup

Two datasets are used to study the performance of our proposals. Dataset GT is a combination of a location-based online social network Gowalla[1] where users share their locations by checking-in and a Twitter dataset containing Twitter messages. A Twitter message is considered as a document and randomly assigned to a location in Gowalla. A fan of a location is the user who checked in at the location. All the users form a friendship network that is undirected and consists of 196,591 nodes and 950,327 edges. Dataset DP is crawled from a popular Chinese online restaurant guide website where users are able to find local restaurants based on personal preferences and share ratings and reviews. Each restaurant has a spatial location and a brief introduction describing its features. A fan of a restaurant is the user who browsed or wrote comments for the restaurant. The social network formed by all users has 722,380 nodes and 1,674,481 edges. Table I provides detailed statistics of the two datasets.

TABLE I. DATA SET STATISTICS

dataset	# of objects	# of distinct words	average # of distinct words per object
GT	1,280,969	1,678,451	14
DP	1,460,000	306,285	107
dataset	# of users	# of edges	average # of fans per object
GT	196,591	950,327	3
DP	722,380	1,674,481	22

We generate 4 query sets in the space of each dataset, in which the number of keywords is 1, 2, 3, and 4, respectively. Each set comprises 100 randomly generated queries. Specifically, to generate a query, we randomly pick an object in the dataset, and take the location of the object as the query location and randomly choose words from the document of the object as the query keywords.

The index structure SNIR-tree is disk resident, and the page size is 8KB. The number of children of a node in the SNIR-tree is computed given the fact that each node occupies a page. This translates to 200 children per node in our implementation. Algorithm 1, denoted as SNIRQP, and the x-hop localized algorithm, denoted as APPRO-x, were implemented in Java, and an Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66 GHz with 4 GB memory was used for the experiments. The Java Virtual Machine Heap is set to 2GB. We report the average elapsed time cost of the queries in each query set. Many layers of cache (e.g., disk driver cache, operating system cache, application cache) exist between a Java application and the physical disk. Rather than measuring physical I/Os from the disk using Java, we report the simulated I/O cost using a simulated LRU buffer that caches 5% nodes in the SNIR-tree.

We study the effect of different parameters and set parameter default values as follows: the number k of requested results is 10; the number of query keywords is 2; parameter α in the social relevance (Equations 2 and 3) is 0.5; parameter x in Equation 3 is 1 (APPRO-x becomes APPRO-1). To compare the retrieved results by SNIRQP and APPRO-x, we adopt the minimum Kendall distance (K_{min}) [13], which is used to measure the distances between top-k lists returned by search engines. It reflects the percentage of the pairs in the opposite order in the two top-k lists. The smaller the value of K_{min}, the more similar the two top-k lists.

[1] http://snap.stanford.edu/data/loc-gowalla.html

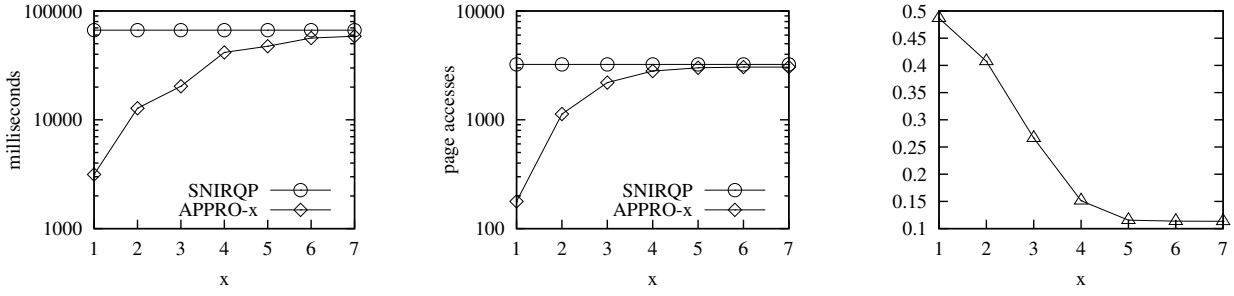
B. Performance Evaluation

We study the proposed methods under various settings, including the performance on different datasets, varying x in Equation 3, varying the number of requested results k, and varying α in the social relevance (Equations 2 and 3). We also report the storage taken up by the index structure SNIR-tree on the two datasets and include the statistics of the tree structure.

Effect of x.
Figures 7 and 8 show the elapsed time, the simulated I/O cost, and the distances (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-x on dataset GT and DP when varying the value of x. When x is assigned small values, e.g., 1 and 2, algorithm APPRO-x outperforms algorithm SNIRQP in terms of the elapsed time and the simulated I/O cost by orders of magnitude. As the value of x increases, the elapsed time and the simulated I/O cost of APPRO-x increase, approaching the performance of SNIRQP. The main reason is that APPRO-x ignores the fans far from the query user when computing the social relevance of objects using a small x. It in some sense reduces the effect of the sparsity of the fan-object space, so that the social relevance of non-leaf entries in algorithm APPRO-x become tight bounds on the social relevance of the entries in their children nodes, comparing with algorithm SNIRQP. Since the social relevance computations in APPRO-x and SNIRQP are different, the two algorithms may retrieve different top-k results. As the value of x increases, the social relevance computed in algorithm APPRO-x become closer to those in algorithm SNIRQP so that the distance (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-x decreases.

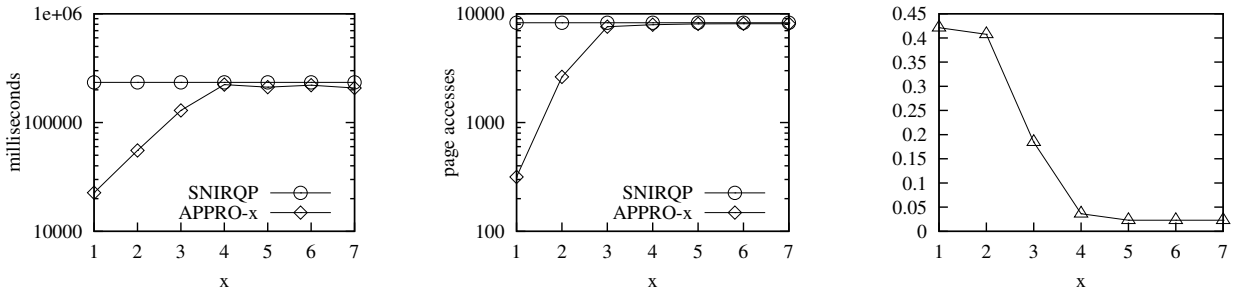
Varying k.
Figures 9 and 10 show the elapsed time, the simulated I/O cost, and the distances (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-1 on dataset GT and DP when varying the number k of requested results. As expected, the elapsed time and the simulated I/O costs of both SNIRQP and APPRO-1 increase slightly as k increases, since more requested objects require more computation costs. Algorithm APPRO-1 outperforms algorithm SNIRQP by orders of magnitude in terms of the elapsed time and the simulated I/O cost for all values of k. The distances (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-1 decreases as k increases, indicating that the social relevance computation (Equation 3) in algorithm APPRO-x only slightly changes the ranking of the top-k result retrieved by SNIRQP. As an example, suppose that SNIRQP returns a ranking p_1, p_2, p_3, p_4 and APPRO-1 retrieves a ranking p_3, p_1, p_2, p_4. The distance K_{min} decreases when k increases from 1 to 4. If a total different ranking p_5, p_6, p_7, p_8 is returned, the distance K_{min} is always large.

Varying α.
Figures 11 and 12 show the elapsed time, the simulated I/O cost, and the distances (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-x on dataset GT and DP when varying parameter α. The elapsed time and the simulated I/O costs of both SNIRQP and APPRO-1 increase as α increases. A small α indicates less impact of the social relevance in the ranking function (Equation 1), while large α results in higher impact of the social relevance in the ranking function. Since the SNIR-tree groups objects based on the spatial proximity, the benefit is significant when α is small. Algorithm APPRO-1



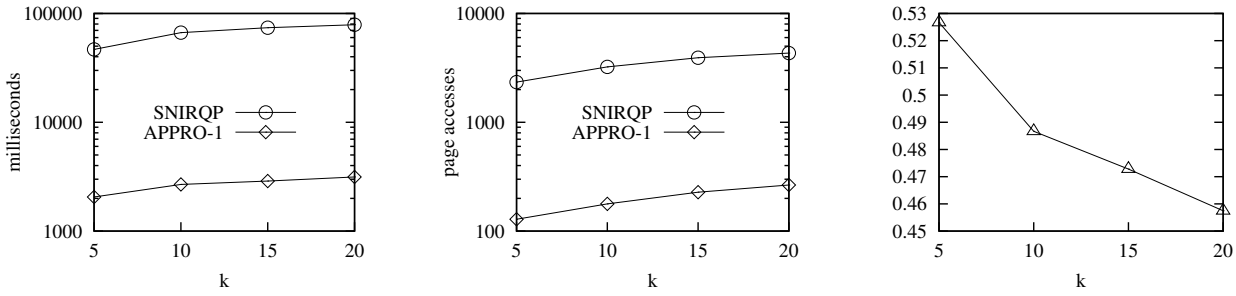
(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 7. Varying x on GT



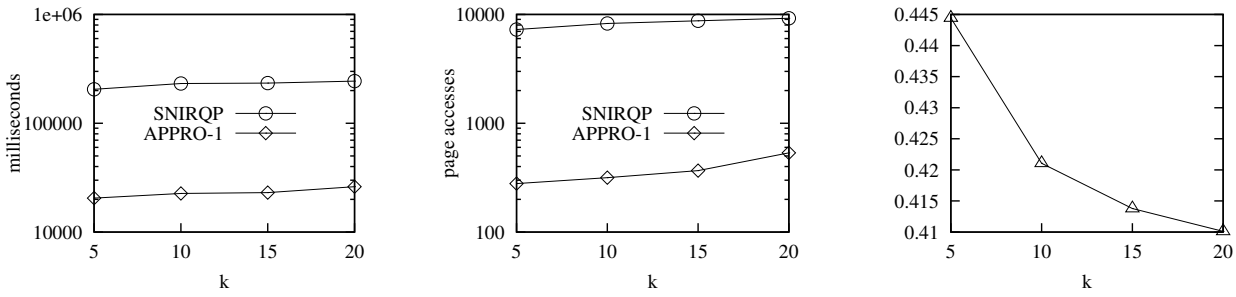
(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 8. Varying x on DP



(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 9. Varying k on GT



(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 10. Varying k on DP

outperforms algorithm SNIRQP significantly in terms of the elapsed time and the simulated I/O cost for all values of α. The distances (K_{min}) between the top-k results retrieved by SNIRQP and APPRO-1 increases as α increases. The reason is that the higher impact (larger α) the social relevance takes in the ranking function, the more different the ranking of the top-k result returned by algorithm APPRO-1 from the top-k result of algorithm SNIRQP. Note that on dataset DP, when $\alpha = 0.1$, algorithm APPRO-1 retrieved the same top-k result ($K_{min} = 0$) as does algorithm SNIRQP.

Space requirement.
Table II shows detailed information of the indexes (SNIR-trees) on the two datasets. The whole index is not buffered by the operating system, given 4GB physical main memory. Datasets GT and DP contain a comparable number of objects, resulting in a comparable number of nodes in their SNIR-trees, while the size of the index on dataset DP is much larger than the index size of dataset GT due to more distinct words and fans per object.

Summary.
The experimental study shows that algorithm APPRO-x outperforms algorithm SNIRQP by orders of magnitude while returning similar top-k results. For the sake of efficiency, we recommend to use small x. Considering the semantics of the social relevance in the ranking function (Equation 1), a small x means that only close friends may help to improve the search results, given the ground truth that close friends tend to have similar interests. Parameter α is used to specify the importance of the social relevance in the ranking function. Users may decide how important the social relevance is for ranking objects.

VI. Related Work

The SkSK query considers social influence in spatial keyword queries. Spatial keyword query processing and social-aware search have been well studied separately. This section covers existing work related to these two kinds of queries.

A. Spatial Keyword Queries

A spatial keyword query retrieves the best object(s) with respect to a given location and a set of keywords. Cao et al. [5] provided an introduction to the subject, and Chen et al. [10] gave an experimental comparison of indexes. Efficient implementation of spatial keyword queries, with varying semantics, has also been studied. Zhou et al. [33] used a hybrid index structure that integrates inverted files and R*-trees for computing both textual and location aware queries. Three different index variations are studied: (1) inverted file and R*-tree double index, (2) first inverted file then R*-tree, (3) first R*-tree then inverted file. All the three variants are used for the processing of spatial keyword queries in two phases, i.e., either first spatial filtering or first keyword filtering. The IR^2-tree [14] is another hybrid index that targets spatial keyword queries. It combines an R-tree with superimposed text signatures. However, it is applicable only when the keywords serve as a Boolean filter. In the IR-tree [12], [19], [25], each node in the R-tree is augmented with inverted files. It supports the ranking of objects based on a weighted sum of spatial distance and text relevance. Recently, other variants of the spatial keyword query have been studied, including the mCK query [31], [32], the Location-aware top-k Prestige-based Text retrieval (LkPT) query [6], the collective spatial keyword query [7], the Reverse Spatial Textual k Nearest Neighbor (RSTkNN) query [20], and the moving top-k spatial keyword query [27]. However, none of the above techniques take into account social influences.

B. Social-aware Search

Utilizing users' social information to improve search results has been studied in the information retrieval community. Carmel et al. [8] investigated personalized social search based on the user's social relations. Search results are re-ranked according to their relations with individuals in the user's social network. SonetRank [18] personalized the Web search results based on the aggregate relevance feedback of the users in similar groups, based on the observation that users in the same group may have similar relevance judgments for queries related to these groups. Xu et al. [28] used the annotation data on del.ico.us and proposed a ranking method based on tag-relevance and match between users' profiles and document tags. Yin et al. [30] studied efficient top-k social web search, i.e., document search considering the social influence, measurement of the affinity between a query user and the publisher of a retrieved document.

VII. Conclusions

This paper studies the social-aware top-k spatial keyword (SkSK) query that enriches the semantics of the spatial keyword query by introducing a new social relevance attribute. We propose a hybrid index structure SNIR-tree that integrates the social information into the IR-tree and develop algorithms for efficient processing of SkSK queries. The index is capable of taking into account social information, text relevance, and location proximity to prune the search space at query time. Results of empirical studies with an implementation of the methods demonstrate that the proposed index and algorithms offer scalability and are capable of excellent performance.

In future work, it is of interest to examine the utility of the SkSK query, i.e., aiming for a result that is as useful as possible for an important use case. However, it is challenging to establish a reliable ground truth for the result of a query. It is necessary to collect real data, including both datasets and query workloads, and determine the usefulness of query results.

Acknowledgment

This work is supported by grants GRF210510, HKBU211512, and HKBU/FRG2/12-13/081.

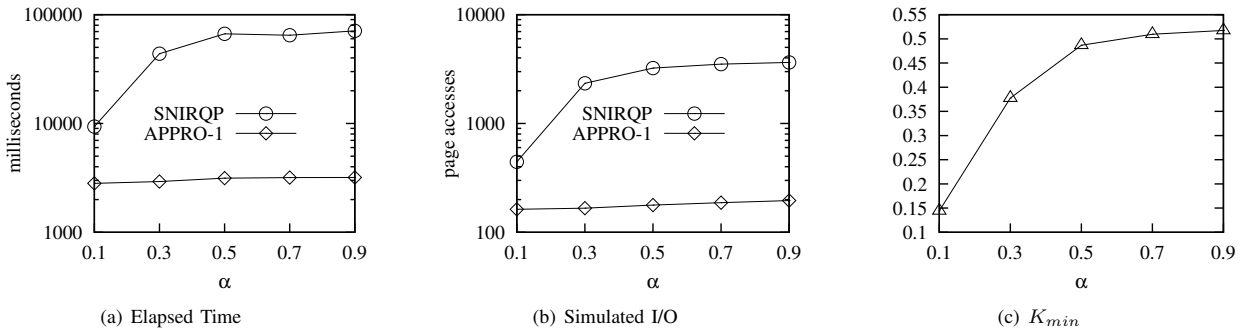
References

[1] S. Amer-Yahia, M. Benedikt, L. V. S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *PVLDB*, 1(1):710–721, 2008.

[2] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing pagerank: damping functions for link-based ranking algorithms. In *SIGIR*, pages 308–315, 2006.

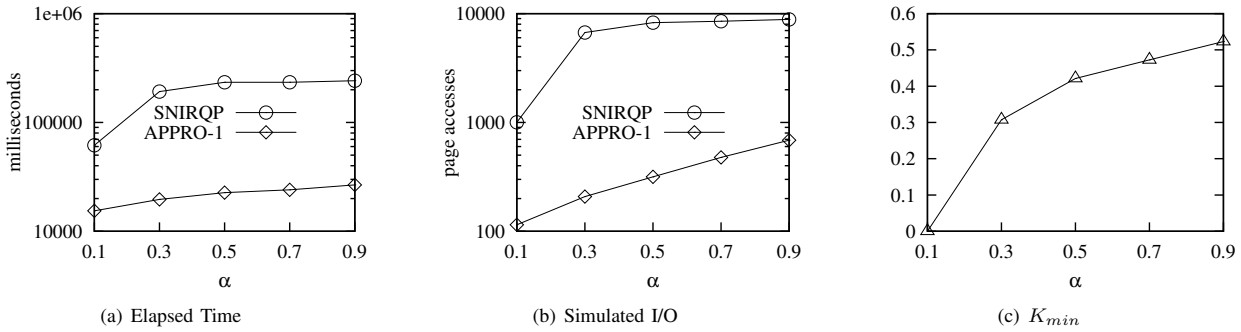
[3] B. Bahmani and A. Goel. Partitioned multi-indexing: bringing order to social search. In *WWW*, pages 399–408, 2012.

[4] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Voting in social networks. In *CIKM*, pages 777–786, 2009.



(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 11. Varying α on GT



(a) Elapsed Time (b) Simulated I/O (c) K_{min}

Fig. 12. Varying α on DP

TABLE II. INDEX STATISTICS

data set	index size	# of levels in the SNIR-tree	# of non-leaf nodes	# of leaf nodes
GT	125.73 GB	3	78	10318
DP	391.62 GB	3	88	11631

[5] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu Spatial keyword querying. In *ER*, pages 16–29, 2012.

[6] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1):373–384, 2010.

[7] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[8] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'El, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *CIKM*, pages 1227–1236, 2009.

[9] E. P. F. Chan and H. Lim. Optimization and evaluation of shortest path queries. *VLDB J.*, 16(3):343–369, 2007.

[10] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. *PVLDB*, 6(3):217–228, 2013.

[11] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, pages 277–288, 2006.

[12] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[13] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discrete Math*, 17(1):134–160, 2003.

[14] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[15] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[16] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[17] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *WWW*, pages 431–440, 2010.

[18] A. Kashyap, R. Amini, and V. Hristidis. SonetRank: leveraging social networks to personalize search. In *CIKM*, pages 2045–2049, 2012.

[19] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. L. Lee, and X. Wang. Ir-tree: an efficient index for geographic document search. *IEEE Trans. Knowl. Data Eng.*, 23(4):585–599, 2011.

[20] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD*, pages 349–360, 2011.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281, 1998.

[22] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222, 2011.

[23] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. de Castro Reis, and B. A. Ribeiro-Neto. Efficient search ranking in social networks. In *CIKM*, pages 563–572, 2007.

[24] F. Wei. Tedi: efficient shortest path query answering on graphs. In *SIGMOD*, pages 99–110, 2010.

[25] D. Wu, G. Cong, and C. S. Jensen. A framework for efficient spatial web object retrieval. *VLDB J.*, 21(6):797–822, 2012.

[26] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Trans. Knowl. Data Eng.*, page to appear, 2011.

[27] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong. Efficient continuously moving top-k spatial keyword query processing. In *ICDE*, pages 541–552, 2011.

[28] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *SIGIR*, pages 155–162, 2008.

[29] M. Ye, X. Liu, and W.-C. Lee. Exploring social influence for rec-

ommendation: a generative model approach. In *SIGIR*, pages 671–680, 2012.

[30] P. Yin, W.-C. Lee, K. C. K. Lee. On top-k social web search. In *CIKM*, pages 1313–1316, 2010.

[31] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.

[32] D. Zhang, B. C. Ooi, and A. K. H. Tung. Locating mapped resources in web 2.0. In *ICDE*, pages 521–532, 2010.

[33] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *CIKM*, pages 155–162, 2005.