Geoinformatica (2012) 16:563–596
DOI 10.1007/s10707-011-0142-7

SKIF-P: a point-based indexing and ranking of web documents for spatial-keyword search

Ali Khodaei · Cyrus Shahabi · Chen Li

Received: 18 January 2011 / Revised: 19 September 2011 /
Accepted: 26 September 2011 / Published online: 22 October 2011

Abstract There is a significant commercial and research interest in location-based web search engines. Given a number of search keywords and one or more locations (geographical points) that a user is interested in, a location-based web search retrieves and ranks the most textually and spatially relevant web pages. In this type of search, both the spatial and textual information should be indexed. Currently, no efficient index structure exists that can handle both the spatial and textual aspects of data simultaneously and accurately. Existing approaches either index space and text separately or use inefficient hybrid index structures with poor performance and inaccurate results. Moreover, most of these approaches cannot accurately rank web-pages based on a combination of space and text and are not easy to integrate into existing search engines. In this paper, we propose a new index structure called Spatial-Keyword Inverted File for Points to handle point-based indexing of web documents in an integrated/efficient manner. To seamlessly find and rank relevant documents, we develop a new distance measure called spatial tf-idf. We propose four variants of spatial-keyword relevance scores and two algorithms to perform top-k searches. As verified by experiments, our proposed techniques outperform existing index structures in terms of search performance and accuracy.

Keywords Geographical search · Spatial databases · Indexing · Ranking · Query processing · Information retrieval

A. Khodaei (✉) · C. Shahabi
Department of Computer Science, University of Southern California,
Los Angeles, CA 90089, USA
e-mail: khodaei@usc.edu

C. Shahabi
e-mail: shahabi@usc.edu

C. Li
Department of Computer Science, University of California-Irvine, Irvine, CA 92697, USA
e-mail: chenli@ics.uci.edu

1 Introduction

There is a large amount of location-based information generated and used by many applications. The Internet is the most popular source of data with location-specific information, such as documents describing events/news at certain locations, Wikipedia pages containing spatial information and images with annotations and information about the places they were taken. Users of such a web-based application often need to query the system by providing requirements on a location as well as keywords in order to find relevant documents, as illustrated by the following example.

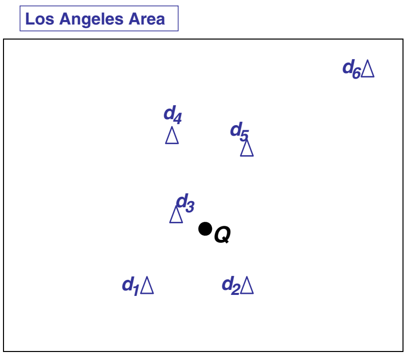
Suppose we have a collection of web objects (documents),[1] and each object contains some text/tags describing an event for a specific location. Web object (document) can be a tweet, an online image or a video. We want to build a system to allow users to search on these objects. Consider a user, Bob, who lives in the central part of Los Angeles and likes to find parks having free concerts around his house. He submits a query to the system with three keywords "`park concert free`" and specifies "`Central Los Angeles`" as the location restriction. Figure 1a shows the location of his query represented as a black circle. Our goal is to find the best documents that are of interest to Bob.

Suppose there are six documents in the repository with locations close to the Central Los Angeles. Figure 1a shows these locations represented as small triangles. In addition, each document has text keywords in its content. Figure 1b shows the frequencies of the three query keywords in these documents. We want to find the most relevant results (documents) to the query. The result cannot be found with a simple keyword-only query since none of the documents may have the actual keywords "`Central Los Angeles`" or even "`Los Angeles`" in them.

In this paper, we show how to support spatial-keyword queries on documents with spatial information. We demonstrate how to rank documents by seamlessly combining spatial and textual features, in order to find highly relevant answers to user queries. In our running example, an interesting question is how to measure the relevance of a document to the query. Intuitively, a document could be of interest to the user if it has at least one of the query keywords, and its location is close to the location mentioned in the query. Document d_6 may not be very relevant to the query, since its location is far from the query location. The other five documents are all much nearer to the query point, and thus could be potentially of more interest to the user. Hence, we need to rank the documents since the user may only be interested in the most relevant documents. However, it is not clear how to measure the relevance of the documents to the user query. For example, it is clear that document d_3 should have a high relevance since its location is very close to the query location, and all three query keywords appear in the document. However, it is not clear how relevant d_2 is to the query, since even though its location is near the query location, it does not have two of the keywords ("`concert free`"). The other documents, d_1, d_4, and d_5, all have the three query keywords, but with different frequencies, and they are at different distances to the query point.

In this paper, we present a ranking method that considers both the spatial proximity of a document to a query and the frequencies of the query keywords in

[1]We use terms *documents* and *objects* interchangeably throughout this paper.



(a) A query with keywords “`park concert free`” and a location “`Los Angeles`” (represented as a black circle). Other points (represented as small triangles) are documents’ locations.

Location	Document	Keywords		
		park	free	concert
Redondo Beach	d_1	5	4	3
Culver City	d_2	3	0	0
Palms	d_3	1	1	1
Santa Monica	d_4	1	1	1
Downtown	d_5	2	2	1
Pasadena	d_6	0	2	1

(b) Documents with location information and keyword frequencies.

Fig. 1 A spatial-keyword query on documents with location information

the document in order to compute a relevance score of the document to the query. We present a new scoring mechanism to calculate the spatial relevance of a document with respect to a query, and propose a method to combine the spatial relevance and the textual relevance.

Given a proper ranking method for documents, a natural question is how to efficiently index and search the location-specific documents. There are several challenges to reach this goal. First, space and text are two totally different data types requiring different index structures. For instance, conventional text engines are set-oriented while location indexes are usually based on two-dimensional and Euclidean spaces. Second, the ranking and search processes should not be separated. Otherwise, the ranking process will rank all the candidate documents (instead of only the relevant documents), rendering the query processing inefficient. Third, the meaning of spatial relevance and textual relevance and a way to combine them using the proposed index structure have to be defined accurately. Finally, it should be easy to integrate the final index structure into existing search engines.

To solve the above problems, we propose a new hybrid index structure called *Spatial-Keyword Inverted File for Points* (“SKIF-P” for short). SKIF-P is an inverted file capable of indexing and searching both textual and spatial data in a similar, integrated manner. With SKIF-P, the space is partitioned into a number of grid cells and each cell is treated similar to a textual keyword. We describe the structure of SKIF-P, and present two efficient algorithms for answering a ranking query using SKIF-P.

A preliminary version of this work appeared in [1], where query and document locations were represented as regions, and we proposed indexing and ranking techniques for that setting. This paper builds on [1] and proposes new indexing and ranking techniques for a setting in which query and document locations are two-dimensional points. For most of the spatial information on the web, locations are either directly specified as two-dimensional points (latitude, longitude) or can be easily converted to that format. Examples of the former are most geo-tagged Web 2.0

objets, such as geo-tagged micro-blogs (e.g. tweets[2]), geo-tagged images (e.g. Flickr[3] images) and geo-tagged videos (e.g. Youtube[4] videos). For the latter, almost all documents (web-pages) containing some spatial information about different places—such as news web-pages (e.g. NY-Times[5]), business listings (e.g. Yelp[6]) and check-in objects (e.g. foursquare[7])—can be geo-coded into two-dimensional geo coordinates (latitude,longitude).[8] In order to define and model the spatial relevance in this new setting (when locations are points and relevance is based on proximity), we define the concept of *spatial decay* and introduce several spatial decay functions. Using the spatial decay functions enable us to define the *spatial relevance* and the spatial-keyword relevance seamlessly and more naturally for the new setting.

Finally, we have conducted an extensive set of experiments to show the efficiency and accuracy of our proposed approaches and the index structure. Using both the real and synthetic datasets we performed a large number of experiments to evaluate the performance of SKIF-P based on several different parameters. When possible, we also compared the performance of SKIF-P with the best existing solutions. We also conducted two separate user studies (based on the real datasets) to evaluate the accuracy of our proposed approaches. Finally, we study the impact of different SKIF-P parameters (e.g., number of cells, type of spatial decay function) on the performance and accuracy of our system.

To summarize, we have the following contributions in this paper:

- We define the problem of ranking queries on documents with spatial and textual information.
- We define a *spatial decay function* and its several variants, to be used seamlessly in our proposed approaches.
- We develop a new scoring method called *spatial tf-idf* to compute the spatial relevance of a document to a query, and combine both spatial relevance and textual relevance for a query.
- We develop an efficient hybrid index structure for indexing both the spatial and textual aspects of the documents. We present two algorithms for answering a ranking query using the structure.
- We have conducted a comprehensive experimental evaluation on real and synthetic datasets to show that our techniques can answer ranking queries efficiently and accurately.

[2] http://www.twitter.com

[3] http://www.flickr.com

[4] http://www.youtube.com

[5] http://www.nytimes.com

[6] http://www.yelp.com.

[7] http://www.foursquare.com

[8] We have to note that although representing locations as points are the current representation of choice on the web, it is not necessarily the most accurate one. For objects with spatial extents, the most accurate presentation is probably a polygon representation. We focus on points in this paper since 1) currently, spatial feature of most objects/documents on the web are represented as points, and 2) we have already showed how to handle cases when the spatial feature is a region in our previous work [1].

The rest of the paper is organized as follows. Section 2 briefly studies the existing index structures. Section 3 defines the problem discussed in this paper along with some preliminary issues. In Section 4 we discuss the concept of spatial decay and introduce several spatial decay functions. Section 5 introduces our ranking method and its several variants. Section 6 presents SKIF-P and its algorithms. Section 7 evaluates our methods through extensive experiments. Finally, Section 8 concludes the paper.

2 Related work

Existing index structures for handling the spatial-keyword queries can be categorized into two broad groups: 1) individual index structures, and 2) hybrid index structures. Individual index structures use one index for each set of features of the data (space or text). The index structures of choice for the spatial data are usually grid, R*-tree or quadtree. For text, inverted files are often used. Using separate index structures, documents satisfying the textual part of the query and documents satisfying the spatial part of the query are retrieved separately using the textual and spatial indexes, respectively. The final result is the merging of the two result sets. An example of this method is the *inverted file and R*-tree double index* presented in [2]. An improved variation of this approach is to filter the results based on one feature first and then use the second index structure on only those results generated from the first step and not on the entire collection. The main problem with the above methods is the fact that each one-feature search usually returns huge number of results.

Hybrid index structures combine the textual and spatial indexes into one index structure. Two basic designs introduced in [2] are *inverted file-R*-tree* and *R*-tree-inverted file*. *Inverted file-R*-tree* is essentially an inverted file on top of R*-trees. With this structure, first the inverted file is built and then R*-trees are built on each inverted list indexing the spatial features of the documents (objects) in the list. For a given spatial-keyword query, the query keywords are filtered using the inverted file and then R*-trees corresponding to those keywords are traversed to search/filter based on the spatial features of the data. This structure performs well only when the number of keywords is very small. Increase in the number of keywords results in traversing multiple R-trees separately and combining the results from those tress, which is very costly. *R*-tree-inverted file* is an R*-tree on top of inverted files. In this structure, an R*-tree is first built for all the documents (objects) locations and then inverted lists are generated for keywords appearing in leaf nodes of the tree. For a given spatial-keyword query, R*-tree's leaf nodes intersecting the query location are retrieved first and then all the inverted lists corresponding to those keywords are traversed. The main disadvantage of this method is in its spatial filtering step, which usually generates many candidate objects.

Two other structures are those presented in [7] and [11]. Both these studies are conceptually similar to the work in [2] with the main difference that they use grid as their spatial index structure. Both also use inverted file as a textual index. With both approaches, spatial and textual search processes are separated and query processing require two stages. Although both approaches use one hybrid index but the spatial and textual search processes are still separated. Both approaches discuss hybrid (spatial and textual) relevance ranking in a very abstract (high level) form. Both fall short to provide a more detailed explanation of how they plan to implement this ranking and how the proposed index structures are used to provide a hybrid relevance ranking. Also, neither of the papers present any type of real experiments (even in the simplest form) to provide and evaluate accuracy of any relevance ranking schema. Also both approaches are based on AND semantics meaning that all the keywords in the query should exist in each result. The main difference between these two approaches ([7] and [11]) is that the focus of the work in [7] is more on system (and algorithmic) issues (more specifically scaling to very large datasets and high query load). Although the work in [11] is very different than what we propose in this paper, as a future direction, the authors of [11] suggested a closer integration of textual and spatial indexing by using spatial cell identifiers as part of the textual index.

Several improved hybrid index structures are introduced more recently. In [3] a hybrid index structure called KR*-tree is proposed. KR*-tree extends R*-tree-inverted file structure by augmenting each node of R-tree with the list of all the keywords appearing in the objects of that subtree. At the query time, KR*-tree is traversed and for each node, not only the spatial intersection with the query region is checked but the node is also checked for the presence of the query keywords. The result to the query are the objects contained in the query region that have all the query keywords (AND semantics). KR*-tree is only good for spatial objects with a small number of keywords. Moreover, KR*-tree cannot be used in the context of location-based web search since documents in the web typically contain large number of keywords. Also the textual relevance ranking (and therefore the spatial-keyword relevance ranking) cannot be supported. Another hybrid index structure called IR^2-tree is presented in [4], which combines R-tree with signature files. With this method, each node in R-tree is augmented with a *signature* representing the union of the keywords (text) of the objects in the subtree rooted at that node. Similar to KR*-tree, IR^2-tree can identify the subtrees that do not contain the query keywords and eliminate them from the search process early on. IR^2-tree is more efficient than KR*-tree since the auxiliary data structure augmented to each node is much smaller (and more efficient) in the IR^2-tree. Similar to the KR*-tree, input data here is a set of spatial objects each associating with number of keywords. Using IR^2-tree, the final result set is a ranked list of objects containing all the query keywords in order of their distances to the query point. IR^2-tree performs better than index structures in [2] and [3] but still has its own shortcomings. At times, the signature files are not able to eliminate the objects not satisfying the query keywords (false hits). This results in loading and reading more objects, which is costly. Furthermore, the performance gets worse when the number of query keywords increases or when the final result is very far from the query point. Another problem with IR^2-tree is that the signature files need to be loaded into memory each time a node is visited. Finally, since there is no intuitive way to use signature files for the textual relevance ranking, IR^2-tree cannot really perform a meaningful ranking. In summary, none of the existing methods mentioned here are designed to support the spatial-keyword relevance ranking. If there is a ranking, it usually is based on the spatial relevance (either distance or overlap). The existing methods use the AND semantics and hence cannot have the results with the partial relevance (object with some but not all the keywords). Also, none of the existing methods use both the spatial and textual aspect of the data simultaneously and in the similar manner. At the best case, they use one feature for pruning the result set and one feature for the actual search (e.g. IR^2-tree uses signature files to prune the data based on the text and uses R-tree to do the actual search). Finally, for all the mentioned methods there are inevitable cases when the actual objects have to be read from the disk and be reevaluated. This happens when the search is based on on feature (e.g. space in IR^2-tree) and pruning on the other feature (e.g. text in IR^2-tree) is not able to prune all the non-qualified objects. In this case, after reaching the candidate object using one feature (e.g. reaching a leaf node in IR^2-tree), the actual object has to be read and rechecked to see if it is qualified based on the other feature (e.g. object's textual description matches the query keyword in IR^2-tree). In other words, in existing methods the index structure by itself is not sufficient for the query processing and access to the actual objects are often needed.

Very recently another hybrid index structure called IR-tree is presented, which also combines R-tree with the inverted files [10]. With this index structure, each node of the R-tree is augmented with an inverted file for the objects contained in the sub-tree rooted at that node. IR-tree is the most similar approach to our work since it considers space and text together. IR-tree is a single index structure requiring only one step to process the query. Also, IR-tree supports ranking of the documents based on both the spatial and textual features (although not very accurately).

Nevertheless, there are some major problems with IR-tree. First, one inverted file needs to be stored and possibly accessed for each node in the tree. For web documents, the total number of documents and the total number of keywords are very large, resulting in huge number of nodes in the tree and also large inverted files for each node. During the search process, the application needs to load the entire inverted file of each visited node into the memory, which causes extra I/Os. Another problem with IR-tree is that during the search process, it often needs to visit few nodes in the tree containing no relevant results. Finally, it is not clear that whether ranking proposed in [10] is an accurate spatial-keyword relevance ranking (see Section 7).

There are many other relevant topics such as approximate keyword search on spatial data [19, 20], *m*-closest keywords (mCK) query (returns the closest objects containing at least *m* keywords) [21], location-aware prestige-based text retrieval (takes into account the inter-relationships among spatial objects) [24], extraction of geographical information [8, 13], geo-coding of documents' locations [12], and geographic crawling [14]. In this paper, we only focus on index structures, relevance ranking, and search algorithms for spatial-keyword queries.

3 Preliminaries

3.1 Problem definition

We assume a collection $D = \{d_0, d_1, ... d_n\}$ of n documents (web pages). Each document d is composed of a set of keywords K_d and a set of locations L_d. Each location is represented by a two dimensional point. We use *the document location* to refer to L_d in this paper.

Spatial-keyword query A spatial-keyword query is defined as $Q = \langle K_q, L_q \rangle$, where L_q is the spatial part of the query specified as one or more two dimensional points and K_q is a set of keywords in the query.

Spatial relevance Spatial relevance between a document d and a query q is defined based on the type of the spatial relationship that exists between L_d and L_q. In [1], we studied the *overlap* relationship while query and document locations were represented as regions (MBRs). Here, we focus on the *proximity* relationship, since query and document locations are points. Subsequently, we define spatial relevance as follows: A document d and a query q are spatially relevant if at least one of the document's locations is within *threshold distance* δ of one of the query's locations, i.e., $distance(L_q, L_d) \leq \delta$. Function *distance* can be any arbitrary distance function such as Euclidian distance, Manhattan (block) distance or road-network distance. The smaller the distance is, the more spatially relevant d and q are. We denote spatial relevance of document d to query q by $sRel_q(d)$.

Textual relevance A document d is textually relevant to the query q if there exists at least one keyword belonging to both d and q, i.e., $K_q \cap K_d \neq \emptyset$. The more keywords q and d has in common, the more they are textually relevant. We represent textual relevance of document d to query q by $kRel_q(d)$. See Section 3.2 for more information regarding textual relevance.

Spatial-keyword relevance A document d is spatial-keyword relevant to the query q if it is both spatially and textually relevant to the query q. Spatial-keyword relevance can be defined by a monotonic scoring function F of textual and spatial relevances. For example, F can be the weighted sum of the spatial and textual relevances:

$$F_q(d) = \begin{cases} \alpha.sRel_q(d) + (1-\alpha).kRel_q(d) & \text{if } sRel_q(d) > 0 \text{ and } kRel_q(d) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

α is a parameter assigning relative weights to spatial and textual relevances. The output of function $F_q(d)$ is the spatial-keyword relevance score of document d and query q, and is denoted by $skRel_q(d)$. In Section 5 we show in details how to calculate spatial-keyword relevance using our proposed index.

Spatial-keyword search A spatial-keyword search identifies all the documents (web pages) that are *spatial-keyword* relevant to q. The result is the top-k most spatial-keyword relevant ranked documents sorted based on documents' spatial-keyword relevance scores. The parameter k is determined by the user.

3.2 Textual relevance

3.2.1 tf-idf Score

Existing textual (keyword) search engines use a *similarity measure* to rank and identify potential (textual) relevant documents. In most keyword queries, a similarity measure is determined by using the following important parameters:

- $f_{d,k}$: the frequency of keyword k in document d
- $\max(f_{d,k})$: maximum value of $f_{d,k}$ over all the keywords in document d

- $\overline{f_{d,k}}$: normalized $f_{d,k}$, which is $\frac{f_{d,k}}{\max(f_{d,k})}$
- f_k: the number of documents containing one or more occurrences of keyword k

Using these values, three monotonicity observations are enforced [5]: (1) less weight is given to the terms that appear in many documents; (2) more weight is given to the terms that appear many times in a document; and (3) less weight is given to the documents that contain many terms. The first property is quantified by measuring the inverse of frequency of keyword k among the documents in the collection. This factor is called *inverse document frequency* or the *idf* score. The second property is quantified by the raw frequency of keyword k inside a document d. This is called *term frequency* or *tf* score, and it describes how well that keyword describes the contents of the document [6, 9]. The third property is quantified by measuring the total number of keywords in the document. This factor is called *document length*.

A simple and very common formula to calculate the similarity between a document d and the query q is shown in Eq. 1.

$$w_{q,k} = \ln\left(1 + \frac{n}{f_k}\right); \; w_{d,k} = \ln(1 + \overline{f_{d,k}});$$

$$W_d = \sqrt{\sum_k w_{d,k}^2}; \quad W_q = \sqrt{\sum_k w_{q,k}^2}; \quad S_{q,d} = \frac{\sum_k w_{d,k}.w_{q,k}}{W_d.W_q}. \tag{1}$$

Variable $w_{d,k}$ captures *the tf score* while variable $w_{q,k}$ captures *the idf score*. W_d represents *document length* and W_q is query length (which can be neglected since it is a constant for a given query). Finally, $S_{q,d}$ is the similarity measure showing how relevant document d and query q are. In this case (textual context) it is the same as $tRel_q(d)$.

3.2.2 Inverted files

Inverted file is the most popular and very efficient data structure for textual query evaluation. Inverted file is a collection of lists, one per keyword, recording the identifiers of the documents containing that keyword [5]. An inverted file consists of two major parts: vocabulary and inverted lists. The vocabulary stores for each keyword k: a count f_k showing number of documents containing k, and a pointer to the corresponding inverted list. The second part of inverted file is a set of inverted lists, each corresponding to a keyword. Each list stores for the corresponding keyword k: identifiers d of documents containing k, and normalized frequencies $\overline{f_{d,k}}$ of term k in document d [5]. A complete inverted file for Example 1 is shown in Fig. 2.

keyword k	f_k	Inverted list for k
`park`	5	$\langle 1,1\rangle\langle 2,1\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle$
`free`	5	$\langle 1,0.8\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle\langle 6,1\rangle$
`concert`	5	$\langle 1,0.6\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,0.5\rangle\langle 6,0.5\rangle$

Fig. 2 Inverted file for Example 1 for each keyword k is composed of the keyword frequency (f_k) and list of pairs, each composed of document id d and normalized keyword frequency $\overline{f_{d,k}}$

4 Spatial decay

According to the first law of geography, “Everything is related to everything else, but near things are more related than distant things.” [25]. Given a location (point), we want to find those points that are *more related* to that location (point). We call the given location(point) *focal point* and the resulting related locations *relevant locations to the focal point* or *relevant locations* in short. The relevance of the relevant locations to the focal point can vary from location to location. Therefore, there is a *weight* assigned to each location. Weight of the focal point itself is always 1 while weight of the non-relevant locations are always 0. Weight of all the other locations (relevant locations) are more than 0 and less than or equal to 1. As we will show in Section 5, in this paper we partition the space into grid cells. As a result, each location essentially corresponds to one cell. In this context, relevant locations are the *grid cells* that are relevant (more related) to the document location (although this can be generalized to other cases and applications) and focal point is the document location.[9] The larger the weight, the more relevant is that cell to the document location. In this section, we show how to find relevant cells to a given focal point and how to calculate the weight of each relevant cell.

In evaluating locations around the focal point, it is common to give less importance to locations (cells) which correspond to the farther locations from the focal point. Intuitively, this reflects the first law of geography mentioned above: nearer locations to the focal point are of more significance while the farther ones are of less significance and can be assigned lower weight or ignored entirely. Several notions of decay functions (and time decay functions) have been used in the literature to capture such characteristics in different data (and temporal data) management applications [22, 23]. Here, we apply and customize some of those functions in our spatial setting and define several *spatial decay functions* to be used in our proposed framework. We consider input items (c_i), which describe all the cells in our setting. We also have focal point FP as another input. FP_c represents the cell associated with the focal point (focal points’s cell).

Definition 1 A *decay function* takes some information about the focal point FP and the ith cell and returns a weight for this cell. We define a function $w(i, FP)$ to be a *decay function* if it satisfies the following properties:

1. $w(i, FP) = 1$ when $c_i = FP_c$ and $0 \leq w(i, FP) \leq 1$ for all $c_i \neq FP_c$.
2. w is monotone non-increasing as *distance* between cells and FP_c increases:

$$distance(c_j, FP_c) \geq distance(c_i, FP_c) \Rightarrow w(j, FP) \leq w(i, FP).$$

3. $w(i, FP) = 0$ when $distance(c_i, FP_c) > \delta$. We call δ *threshold* value and is used to prune the locations (points) that are not very related (relevant) to the focal point.

In this paper, we focus on decay functions of a certain form: where the weight of a cell can be written as a function of its distance *dist*, where the *dist* for cell *i*

[9]For simplicity, we assume that each document has only one location. Multiple locations can be easily handled by using the same methods multiple times—once for each focal point.

$(c_i \neq FP_c)$ is simply $dist = distance(c_i, FP_c)$. Here and in all the other references in this paper, *distance* can be any distance function in the metric space as long as it satisfies the three main distance properties: 1) non-negativity: the distance between distinct points is positive, 2) symmetry: the distance from x to y is the same as the distance from y to x, and 3) triangle inequality: the distance from x to z via y is at least as great as from x to z directly. Euclidian distance, block (Manhattan) distance and road-network distance are valid examples of such a distance function (throughout this paper, we will use the Euclidian distance function from the center of cells).

Definition 2 A *spatial decay function* is defined by a positive monotone non-increasing function $f()$ so that the weight of the ith cell to focal point FP is given by:

$$w(i, FP) = \frac{f(distance(c_i, FP_c))}{f(distance(FP_c - FP_c))} = \frac{f(distance(c_i, FP_c))}{f(0)} \tag{2}$$

The denominator in the equation is to normalize the weigh and make the first property of Definition 1 valid. Different choices of function f generate several interesting spatial decay functions. We study three of these decay functions here and report the effect of each one in Section 7.

4.1 Windows decay

With the windows decay, all the cells whose distance from the focal point is less than the threshold value δ are considered and more importantly are treated the same (i.e., they have the same weight). Setting the "window size" parameter equal to the threshold parameter δ, the function $f(dist) = 1$ for $dist \leq \delta$ and $f(dist) = 0$ for $dist > \delta$ (Fig. 3).

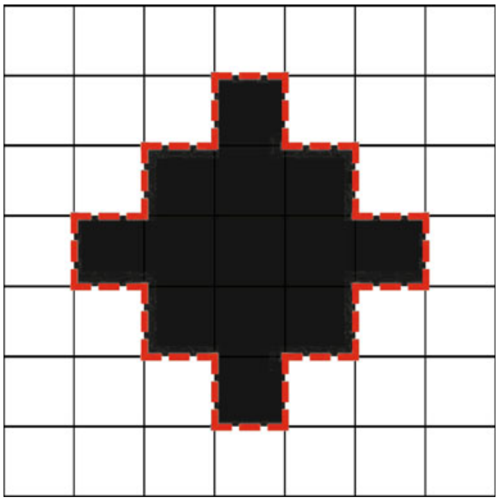


Fig. 3 Example of windows decay with δ equal to 2 cells (Euclidian) and focal point at the center of the grid

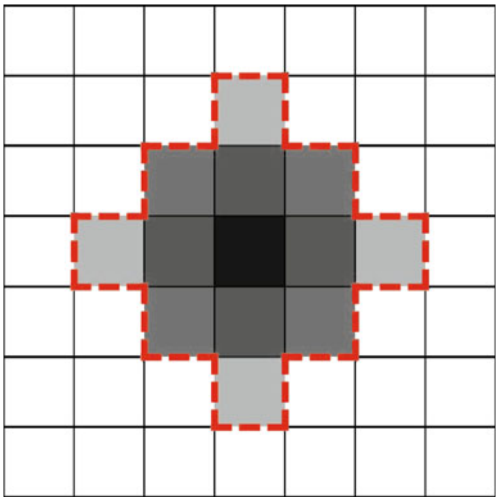


Fig. 4 Example of polynomial decay with δ equal to 2 cells (Euclidian) and focal point at the center of the grid

4.2 Polynomial decay

Most often, treating cells (locations) as either relevant or not is not precise enough. Most of the time, we need a more complicated and fuzzy weighting mechanism. That is the reason polynomial and exponential decay functions are defined. Spatial polynomial decay is defined as $f(dist) = (dist + 1)^{-\gamma}$, for some $\gamma > 0$. Here, 1 is added to *dist* to ensure that $f(0) = 1$. We can also write the equation as: $f(a) = exp(-\gamma ln(dist + 1))$. $f(dist)$ is still zero for $dist > \delta$ (Fig. 4).

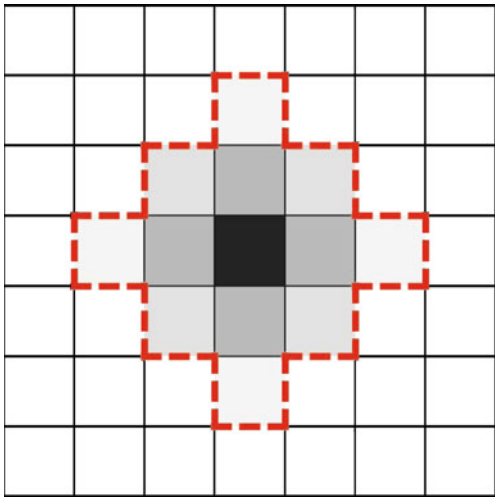


Fig. 5 Example of exponential decay with δ equal to 2 cells (Euclidian) and focal point at the center of the grid

4.3 Exponential decay

Sometimes, polynomial decay is too slow (weight changes are not very significant) and a faster decay function is needed. Spatial exponential decay is defined as $f(dist) = exp(-\lambda \times dist)$ for $\lambda > 0$. Again, $f(dist)$ is zero for $dist > \delta$ (Fig. 5).

5 Seamless spatial-keyword ranking

In this section, we define new scoring mechanism to calculate the spatial relevance and spatial-keyword relevance scores. Following the same intuitions and concepts used in regular (textual) searches, we define new concepts and parameters for spatial data. Most notably, inspired by tf-idf in textual context, we define a new scoring mechanism called *spatial tf-idf* for the spatial context. Using (textual) tf-idf scores and spatial tf-idf scores, the spatial-keyword relevance is defined and can be used to rank the documents based on both the spatial and textual aspects of the data, simultaneously and efficiently. We discuss two different approaches to calculate the spatial-keyword relevance using the spatial tf-idf score. Several variants of the final similarity measure is also presented.

5.1 Spatial tf-idf

In order to be able to use the analogous ideas used in the regular tf-idf score, we need to treat spatial data similar to textual data. Most importantly, we need to represent space which is coherent and continuous in nature, as disjunct and set-oriented units of data—similar to the textual keywords. Hence, we partition the space into grid cells and assign unique identifiers to each cell. Therefore, each location in document can be associated with a cell identifier. Since we are using proximity as our main spatial query type, these cells are defined as the nearby cells to the document location. With spatial tf-idf, the closeness of a cell with the document location is analogous to the existence of a keyword in document with tf-idf. However, knowing the nearby cells is not enough. We need to know how well a cell describes the spatial content of the document. We use the distance between each cell and the document location to provide a measure of how well that cell describes the document. Analogous to *frequency of keyword k in document d*, we define *frequency of cell c in document d*. We represent *frequency of cell c in document d* by $f_{d,c}$ and set its value equal to the value of a spatial decay function (Definition 2 in Section 4) where focal point FP is the document d's location (L_d) and i is the index of cell c. The value of $f_{d,c}$ is monotone non-increasing as the distance between the document location L_d and the cell c increases. Similar to the frequency of a keyword which describes how well the keyword describes the documents textual contents (K_d), the frequency of a cell describes how well the cell describes the documents spatial contents (L_d). The smaller the distance, the better this cell describes the document location and vice-versa. Note that the different variations of the spatial decay function generate different values for $f_{d,c}$.

Now we can define the following parameters analogous to those of Section 3.2:

- $f_{d,c}$: the frequency of cell c in document d
- $\max(f_{d,c})$: maximum value of $f_{d,c}$ over all the cells in document d

- $\overline{f_{d,c}}$: normalized $f_{d,c}$, which is $\frac{f_{d,c}}{\max(f_{d,c})}$
- f_c: the number of documents containing one or more occurrences of cell c

Using the above parameters, we revisit three monotonicity properties discussed in Section 3.2, this time in spatial context: (1) less weight is given to cells that appear in many documents; (2) more weight is given to cells that are closer to the document location; and (3) less weight is given to documents that contain many cells.

The first property is quantified by measuring the inverse of frequency of a cell c among the documents in the collection. We call this *spatial inverse document frequency* or idf_s score. The second property is quantified by the frequency of cell c in document d (as defined earlier). This is called *spatial term frequency* or tf_s score and describes how well that cell describes the document spatial contents (i.e. L_d). The third property is quantified by measuring the total number of cells in the document (remember that documents can have multiple locations). This factor is called *document spatial length*.

Among the above properties, properties (2) and (3) are more intuitive. Property (2) states that more weight should be given to the cells closer to the document location. The smaller the distance, the better that cell describes the document location. For example, in Fig. 6, cell c_9 better describes the document d_6 than cell c_8 and cell c_8 better describes the document d_6 than cell c_3. Property (3) states that less wight should be given to those documents whose locations cover more cells. As we defined in our problem definition, each document can have multiple locations and hence multiple cells will be associated with it. Assuming all the other parameters are equal, a document with a smaller coverage (fewer number of cells) should get a higher weight than a document with a larger coverage. Assume two documents one containing locations of all the cities in the world and the other one containing only one location (city). When searching for/near that city, the second document should be assigned more weight since it is a better representative of the queried city/location than the first document. This is analogous to the fact that in textual context, more weight is given to the documents that contain fewer keywords.

Contrary to properties (2) and (3), property (1) is not very intuitive. It states that less weight is given to the cells appearing in more documents. In the textual context,

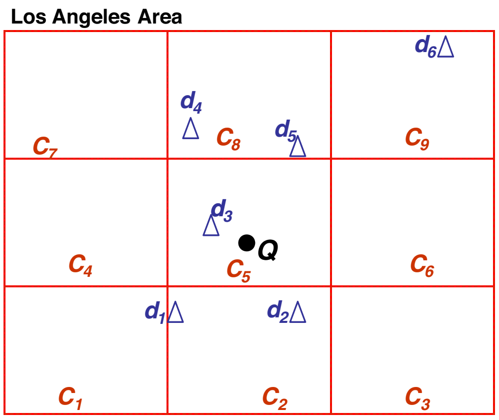


Fig. 6 Example 1 on the grid

the idf score is a weighting factor determining the importance of each keyword independent of the query. It assigns more weight to keywords appearing in fewer documents, since those are more *meaningful* keywords. However, the definition of *meaningful cell* is not very clear in the spatial context. A popular cell (location)—a cell near to many documents—is a very meaningful cell for some users/applications, while for some others, a distinctive cell (location)—cell near to few documents—is more meaningful. In Example 1, one user may look for more popular locations for parks while another user may be interested in a less crowded, more private location. To cover both cases, we define spatial idf of cell c in two different ways: inverse of frequency of a cell c among the documents (*inverted* idf_s) and direct frequency of a cell c among the documents (*direct* idf_s).

5.2 Spatial-keyword relevance

In this section, we introduce two novel approaches for calculating spatial-keyword relevance between a document d and a query q. With the *single-score* approach, one similarity measure and one document length is used to combine the spatial relevance and textual relevance into one equation. With the *double-score* approach, spatial and textual relevance are calculated separately, using two document lengths, one for each relevance. Thus a new spatial similarity measure analogous to the textual similarity measure is defined. Both approaches can use the parameter α to assign relative weights.

5.2.1 Single-score approach

After partitioning each document location to a set of cells, defining the spatial tf-idf score and creating one document spatial length for each document location, the cells are ready to be treated in a similar manner to the keywords. We define *term* as the smallest unit of data describing each document which is either a keyword or a cell. If we represent keywords associated with the document d by K_d and the cells associated with the same document by C_d, then the set of terms associated with document d is represented by T_d and defined as follows: $T_d = K_d \cup C_d$.

Simply stated, the document's terms are the union of the document's keywords and cells. For Instance, in Example 1: $T_{d_1} = \{\texttt{park}, \texttt{free}, \texttt{concert}, \texttt{c}_2\}$ when $\delta = 0$ or $T_{d_1} = \{\texttt{park}, \texttt{free}, \texttt{concert}, \texttt{c}_1, \texttt{c}_2, \texttt{c}_3, \texttt{c}_5\}$ when $\delta = 1$ (see Figs. 1b and 6). In order to be able to define a single similarity measure capturing both the textual and spatial relevances, we define the following parameters:

- $\underline{f_{d,t}}$: the frequency of term t in document d
- $\overline{f_{d,t}}$: the normalized frequency of term t in document d
- f_t: the number of documents containing occurrences of term t

where each parameter gets its value from the corresponding parameter in the space or text domain (based on term type). For instance, value of $f_{d,t}$ is equal to $f_{d,k}$ when term is keyword and to $f_{d,c}$ when term is cell. Having defined these new parameters, we can now easily redefine Eq. 1, this time with terms instead of keywords. This is a new formulation capturing the keywords (textual relevance) and the cells (spatial relevance) in a unified manner.

$$
\begin{aligned}
w_{q,t} &= \begin{cases} (1-\alpha).\ln\left(1+\dfrac{n}{f_t}\right) & \text{if } t \text{ is keyword} \\ \alpha.w_{q,c} & \text{if } t \text{ is a cell} \end{cases} ; \\
w_{d,t} &= \begin{cases} (1-\alpha).\ln(1+\overline{f_{d,t}}) & \text{if } t \text{ is keyword} \\ \alpha.\ln(1+\overline{f_{d,t}}) & \text{if } t \text{ is cell} \end{cases} ; \\
\widehat{W_d} &= \sqrt{\sum_t w_{d,t}^2}; \qquad \widehat{W_q} = \sqrt{\sum_t w_{q,t}^2}; \\
\widehat{S}_{q,d} &= \frac{\sum_t w_{d,t}.w_{q,t}}{\widehat{W_d}.\widehat{W_q}}.
\end{aligned} \tag{3}
$$

The variable $w_{d,t}$ captures the *spatial-keyword term frequency* score (tf_{sk}). The variable $w_{q,t}$ captures the *spatial-keyword inverted document frequency* (idf_{sk}). Parameter α is integrated into the weighting scheme to capture the weighted relevance of space versus text. $\widehat{W_d}$ represents *spatial-keyword document length* and $\widehat{W_q}$ is (spatial-keyword) query length. Finally, $\widehat{S}_{q,d}$ is the similarity measure showing how *spatial-keyword relevant* document d is to query q.

5.2.2 Double-score approach

In the *single-score* approach, keywords and cells are treated in exactly the same manner. Keywords and cells tf and idf scores are used in one equation and one similarity measure ($\widehat{S}_{q,d}$) using one document length ($\widehat{W_d}$) is used to calculate the final relevance score. There might be cases when most of the documents in the collection contain many locations but very few keywords (or the opposite). In this situation, it is better to calculate the textual and spatial relevance scores separately. Hence, we discuss another approach to calculate the similarity measure between document d and query q in the spatial-keyword context. One can first calculate the spatial relevance and the textual relevance of document d and query q independently and then use an aggregation function to compute the overall spatial-keyword relevance score. Using the spatial tf-idf parameters and the definitions, we calculate the spatial similarity measure between document d and query q analogous to the textual similarity measure as follows:

$$
\begin{aligned}
w_{q,c} &= \begin{cases} \ln\left(1+\dfrac{n}{f_c}\right) & \text{if inverted document frequency} \\ \ln\left(1+\dfrac{f_c}{n}\right) & \text{if direct document frequency} \end{cases} ; \quad w_{d,c} = \ln(1+\overline{f_{d,c}}); \\
W'_d &= \sqrt{\sum_c w_{d,c}^2}; \qquad W'_q = \sqrt{\sum_c w_{q,c}^2}; \\
S'_{q,d} &= \frac{\sum_c w_{d,c}.w_{q,c}}{W'_d.W'_q}.
\end{aligned} \tag{4}
$$

where $S'_{q,d}$ is the spatial similarity measure between document d and query q. This value captures the spatial relevance $sRel_q(d)$ defined in Section 3.1.

After calculating the spatial relevance using the above equation and computing the textual relevance using Eq. 1, the aggregation function F can be used to calculate the final spatial-keyword relevance. More formally: $skRel_q(d) = \alpha.S'_{q,d} + (1-\alpha).S_{q,d}$.

Variants We conclude this section by summarizing possible variants of the spatial-keyword relevance score. We defined two different approaches to calculate the spatial-keyword relevance scores. We also introduced two different ways to define the *spatial idf* factor score. Combining our two main approaches with the two definitions of the spatial idf score yields four different variants for our final similarity measure:

1. Single-Score with Inverted document frequency (SSI)
   Where $skRel_q(d) = \widehat{S_{q,d}}$ and $w_{q,c} = \ln\left(1 + \frac{n}{f_c}\right)$
2. Single-Score with Direct document frequency (SSD)
   Where $skRel_q(d) = \widehat{S_{q,d}}$ and $w_{q,c} = \ln\left(1 + \frac{f_c}{n}\right)$
3. Double-Score with Inverted document frequency (DSI)
   Where $skRel_q(d) = \alpha.sRel_q(d) + (1-\alpha).kRel_q(d)$ and $w_{q,c} = \ln\left(1 + \frac{n}{f_c}\right)$
4. Double-Score with Direct document frequency (DSD)
   Where $skRel_q(d) = \alpha.sRel_q(d) + (1-\alpha).kRel_q(d)$ and $w_{q,c} = \ln\left(1 + \frac{f_c}{n}\right)$

6 Spatial-Keyword Inverted File

Spatial-Keyword Inverted File (SKIF-P) is an inverted file capable of indexing and searching both the textual and spatial data in a similar, integrated manner using a single data structure. In this section, we first describe the structure of SKIF-P and the information it stores. Next, we show how spatial-keyword query evaluation is performed using SKIF-P. Two algorithms corresponding to our two approaches are presented. Finally, we discuss briefly how SKIF-P can be extended to more general cases.

6.1 SKIF-P structure

Since SKIF-P is an inverted file, its structure is very similar to the structure of the regular inverted files. SKIF-P consists of two parts: vocabulary and inverted lists. The vocabulary contains all the *terms* in the system which includes all the (textual) keywords and cells (cell identifiers). For each distinct term, three values are stored in the vocabulary: 1) f_t representing the number of the documents containing the term t, 2) a pointer to the corresponding inverted list and 3) the type of the term t which is used to help calculate the tf and idf scores. The second component of SKIF-P is a set of inverted lists each corresponding to a term. For the corresponding term t, each list stores the following values: identifiers of the documents containing term t and the normalized frequencies of term t for each document d. The latter is represented by $\overline{f_{d,t}}$. Figure 6 redraws the Example 1 on the grid and Fig. 7 shows the complete SKIF-P for Example 1. To calculate the values of this example, we used a *polynomial* decay function with $\gamma = 1.8$. Also we used *Euclidian* distance function with value of δ set

Fig. 7 Spatial-keyword inverted file for Example 1 (for $\delta = 0$). The entry for each term t is composed of the term frequency (f_t) and list of pairs, each composed of document id d and normalized term frequency $\overline{f_{d,t}}$

term t	f_t	type	Spatial-Keyword Inverted List for t
`park`	5	1	$\langle 1,1\rangle\langle 2,1\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle$
`free`	5	1	$\langle 1,0.8\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle\langle 6,1\rangle$
`concert`	5	1	$\langle 1,0.6\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,0.5\rangle\langle 6,0.5\rangle$
c_2	2	0	$\langle 1,1\rangle\langle 2,1\rangle$
c_5	1	0	$\langle 3,1\rangle$
c_8	2	0	$\langle 4,1\rangle\langle 5,1\rangle$
c_9	1	0	$\langle 6,1\rangle$

at 0 (since $\delta = 0$, values of $\overline{f_{d,t}}$ are all equal to 1. For other values of δ, the inverted index and its values will be more complex).

6.2 Query processing

As discussed in Section 3.1, the spatial-keyword query consists of two parts: the query keywords K_q and the query location L_q. To process spatial-keyword queries, we first need to convert L_q to a set of cells C_q. C_q is the set of cells near the document location L_q (calculated using spatial decay function and threshold parameter). After calculating C_q, we define the set of terms associated with each query by T_q as follows: $T_q = K_q \cup C_q$.

Algorithms 1 and 2 show the algorithms to perform top-k spatial-keyword search using SKIF-P for the single-score and the double-score approaches respectively. Both the algorithms are very similar. Accumulators are used to store the partial similarity scores. The main difference is that Algorithm 1 uses one accumulator A_d while the Algorithm 2 uses two accumulators A_d and A'_d. After all the query terms are processed, similarity scores $\widehat{S}_{q,d}$, $S_{q,d}$ and $S'_{q,d}$ are derived by dividing each accumulator value by the corresponding value $\widehat{W_d}$, W_d and W'_d respectively (first one used in the single-score algorithm while the last two are used in double-score algorithm). Finally, the k largest documents are identified and will be returned to the user.

In the single-score approach (Algorithm 1), we assign one accumulator for each document d which is denoted by A_d. Partial similarity scores are stored in these

Algorithm 1 top-k spatial-keyword search using SKIF-P, single-score approach

```
 1: Allocate an accumulator $A_d$ for each document $d$
 2: Set $A_d \leftarrow 0$
 3: for each query term $t$ in $q$ do
 4:     Calculate $w_{q,t}$ and fetch the inverted list for $t$
 5:     for each pair $< d, \overline{f_{d,t}} >$ in the inverted list do
 6:         Calculate $w_{d,t}$
 7:         Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
 8: Read the array of $\widehat{W_d}$ values
 9: for each $A_d > 0$ do
10:     Set $\widehat{S_d} \leftarrow A_d \div \widehat{W_d}$
11: Identify the $k$ greatest $\widehat{S_d}$ values and returns the corresponding documents
```

Algorithm 2 top-k spatial-keyword search using SKIF-P, double-score approach

```
1: Allocate two accumulators $A_d$ and $A'_d$ for each document $d$
2: Set $A_d \leftarrow 0$
3: Set $A'_d \leftarrow 0$
4: for each query term $t$ in $q$ do
5:     Calculate $w_{q,t}$ and fetch the inverted list for $t$
6:     for each pair $< d, \overline{f_{d,t}} >$ in the inverted list do
7:         Calculate $w_{d,t}$
8:         if type of $t$ is a keyword then
9:             Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
10:        else
11:            Set $A'_d \leftarrow A'_d + w_{q,t} \times w_{d,t}$
12: Read the array of $W_d$ values
13: for each $A_d > 0$ do
14:     Set $S_d \leftarrow A_d \div W_d$
15: Read the array of $W'_d$ values
16: for each $A'_d > 0$ do
17:     Set $S'_d \leftarrow A'_d \div W'_d$
18:     if $A_d > 0$ then
19:         $\widehat{S_d} = \alpha.S'_d + (1-\alpha).S_d$
20: Identify the $k$ greatest $\widehat{S_d}$ values
```

accumulators. Initially, all the accumulators have a value of zero (e.g., similarity of zero). The query terms are processed one at a time and for term t, the accumulator A_d for each document d included in the t's inverted list is increased by the contribution of t to the similarity of d and q. After all query terms are processed, similarity scores $\widehat{S_{q,d}}$ are derived by dividing each accumulator value by the corresponding value $\widehat{W_d}$. Finally, the k largest documents are identified and will be returned to the user.

In the double-score approach (Algorithm 2), two accumulators are assigned to each document: A_d and A'_d. The partial textual similarity score is stored in A_d while the partial spatial similarity score is stored in A'_d. Initially, both accumulators are empty (zero score). Again, terms are processed one at a time and for each term t and for each document d included in t's inverted list, values of A_d and A'_d are increased by the contribution of term t to the textual and spatial similarity of document d and q, respectively. After processing all the query terms, spatial similarity scores $S'_{q,d}$ are calculated by dividing each A'_d to its corresponding W'_d. In addition, textual similarity scores $S_{q,d}$ are computed by dividing the values of A_d accumulators to the corresponding W_d values. Finally, for each document, if both similarity scores are larger than zero, the final similarity score is calculated as the weighted sum of these two scores.

Algorithms 1 and 2 present the general framework for query processing using our proposed data structure and scoring methods. Since both algorithms are built on top of inverted files, they can be easily improved and scaled using numerous optimization techniques available for inverted files. Few techniques that can be easily applied to SKIF-P are: accumulator thresholding [5], frequency-sorted indexes and impact-ordered inverted lists [5] and various compression schemas [15]. Also existing frameworks to support distributed and parallel computing (e.g., map-reduce framework) can be easily used for our algorithms. Since optimization of inverted files is not our contribution, we do not discuss such optimizations here.

In summary, using our proposed approaches with SKIF-P, the relevance ranking is based on both the space and text and partial relevance is also supported. The search and ranking processes are combined together effectively and more importantly, the spatial and textual aspects of data are handled simultaneously and in a similar manner. SKIF-P by itself (with the addition of accumulators) is sufficient for the query processing and extra access to the actual documents (objects) is not needed.

6.3 Generalization

In this section we briefly show how we can extend SKIF-P into more general cases.

Distributed architecture When a large number of documents is involved or when the total number of keywords and/or locations is huge, it is more practical and efficient to have a distributed system with many computers than just a single machine. In our context, distribution means splitting the document collections and/or its index structure (SKIF-P) across multiple machines and answering the queries by synthesizing the results from various components. Two main architectures exist to make a (inverted-index based) retrieval system distributed: document-distributed architectures and term-distributed architectures [18]. Both designs can be easily applied to our proposed approaches and index structure. With document-distributed architecture, the document collection is partitioned into several sub-collections and each sub-collection is assigned to a machine. No change is needed to use this architecture for our proposed system. In term-distributed architecture, the index is split into components by partitioning the vocabulary. With SKIF-P, since we treat cells similar to keywords, we can also partition our vocabulary (cells and keywords) into several components. More interestingly, we can take advantage of the fact that nearby cells are usually queried together and hence partition the cells based on their spatial proximity. In other words, each component contains cells for one specific area of the geographical map. Obviously, this will improve the performance of the system significantly (fewer partitions need to be retrieved and evaluated).

Weights When querying the system, there are two types of weights users may want to manipulate: 1) assigning different weights to the spatial and textual relevance, and 2) assigning different weights to different terms in the query. For the first scenario, we have used the parameter α in this paper. SKIF-P can also support assigning different weights for different terms. There are several existing methods to solve this problem for textual keywords. Since we are treating cells similar to keywords, those methods can also be applied to the spatial cells. As one possible solution, we define *query term weights* $\alpha_{q,k}$ and $\alpha_{q,c}$ as the weight of keyword k in query q and the weight of cell c in query q, respectively. By multiplying $w_{d,k}$ and $w_{d,c}$ values by $\alpha_{q,k}$ and $\alpha_{q,c}$, respectively, query term weights are integrated into the relevance scores. This opens up a wide possibility of sophisticated queries for a user. For instance, the user can search for all the documents regarding "`Andre Agassi`" plus the following spatial locations in the order of importance: New York, Paris, London and Sydney. This query (which was never possible before) shows that the user is more interested in results regarding Andre Agassi history in the US OPEN (NY) than his results in the French Open (Paris) and so on.

Leveraging existing search engines One of the most practical advantages of the proposed approach is the fact that it can be integrated into the existing search engines easily and seamlessly. Since the structure of SKIF-P is very similar to the structure of the regular inverted files, the same techniques used in regular search engines (built on inverted files) can be applied for our location-based search engine (built on SKIF-P). Structure wise, the main difference is in using SKIF-P instead of the regular inverted files. This essentially translates to using a larger vocabulary (combination of cells and keywords instead of only keywords). Average size of inverted lists would not change since it only depends on the total number of the documents and that is fixed. This is very promising because the cost of existing search engines is dominated by the cost of traversing the inverted lists and not the size of the vocabulary [26]. The easy integration of our approach into the existing search engines is not only very beneficial for current search engines but also enables us to optimize SKIF-P using a body of work that exists in this field. For example, *compression* techniques are very popular for inverted files [5, 15]. Since the structure of the inverted lists are identical with both SKIF-P and regular inverted files, no change is needed to apply the same compression techniques on SKIF-P. More interestingly, some of the optimization techniques seem to work better on SKIF-P. For instance, *caching* is another technique used in existing search engines. It is trivial to see that with SKIF-P, by caching the inverted lists for the cells nearby the current query cell, we can improve the query performance significantly. It is very likely that nearby cells queried together or very close to each other.

7 Experiments

In this section, we experimentally evaluate the performance and accuracy of SKIF-P. Comparison is done with the most efficient proposed solutions: *MIR*^{2}*-tree* [4] and *CDIR-tree* [10] which are optimized versions of IR^2-tree and IR-tree, respectively.

Our experiments are run on three datasets, two real datasets and one synthetic dataset. Dataset properties are summarized in Table 1. DATASET1 is generated from a real world online web application called Shoah Foundation Visual History Archive (http://college.usc.edu/vhi/). Each document (testimony) is tagged with a set of textual and spatial keywords describing the content of the testimony. In preparing DATASET1, we extracted location names (spatial keywords) from all the testimonies and geo-coded the location names into spatial points using Yahoo! Place-

Table 1 Dataset details

Dataset	Total # of documents	Average # of unique keywords per document	Total # of unique keywords	Total # of keywords
DATASET1	19,841	64	31,721	1,269,824
DATASET2	250,000	230	50,000	57,500,000
DATASET3	8,964	11	2,340	109,604

maker.[10] We run our experiments on all the documents in US. For DATASET1, we partition the space into 100 km $\times$ 100 km cells. DATASET2 is generated synthetically. A set of keywords (from 1 to 500) and one location (point) are assigned randomly to each document. Space is partitioned into 40×40 cells. The threshold value δ is set to 2 cells and we use *polynomial* decay function with *gamma* set to 1.8 (standard value). Experiments with different cell sizes and also different δ values are also performed and the results are shown later in this section. The documents' keywords and locations are uniformly distributed. Finally, DATASET3 is generated from some of the geo-tagged images from the online photo-sharing website Flickr using Flickr API.[11] We chose random (geo-tagged) documents inside California. Again, each document (image) is tagged with a set of textual keywords and one (point) location.

Each query contains 1 to 4 randomly generated keywords and one two-dimensional point. Each query *round* consists of 100 queries. All three structures are disk-resident and the page size is fixed at 4KB. MIR^2-tree and CDIR-Tree implementations are the same as in [4] and [10],[12] respectively. Experiments were run on a machine with an Intel Core2 Duo 3.16 GHz CPU and with 4GB main memory.

7.1 Performance

First, we evaluate the performance of SKIF-P based on the most important metrics (number of keywords, number of results and α) and compare the results with both CDIR-tree and MIR^2-tree. We use DATASET1 and DATASET2 for this performance evaluation.[13]

Number of keywords With the first set of experiments, we evaluate the impact of the number of keywords in each query $|K_q|$ on the query cost. In this set of experiments, we vary $|K_q|$ from 1 to 4 while fixing k at 10 and α at 0.5. For each method, we report the average query cost in processing each round. The results are shown in Fig. 8a and b. For almost all the cases, SKIF-P outperforms both MIR^2-tree and CDIR-tree. While for all the approaches, the query cost increases as $|K_q|$ grows, the growth rate for SKIF-P is very marginal. While the I/O costs of CDIR-tree and MIR^2-tree increase by a factor of 15 and 8, respectively, SKIF-P's query cost does not even double when the number of keywords grows from 1 to 4. Both CDIR-tree and MIR^2-tree would perform even worse if $|K_q|$ increase further. This is because with IR^2-tree, by increasing the number of keywords, fewer documents will contain all the keywords and hence more documents need to be searched (this also increases the number of false hits). With CDIR-tree, when query contains more keywords, the textual relevance of the query to each node of CDIR-tree is very similar, which makes the textual relevance pruning less effective. Therefore, both approaches need

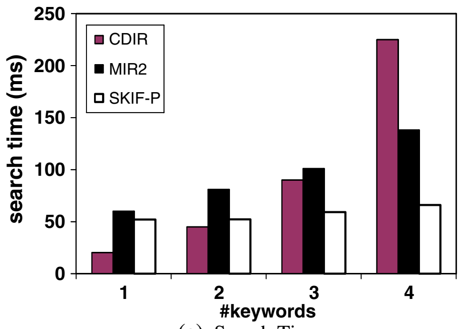
[10] http://developer.yahoo.com/geo/placemaker/

[11] http://www.flickr.com/services/api/

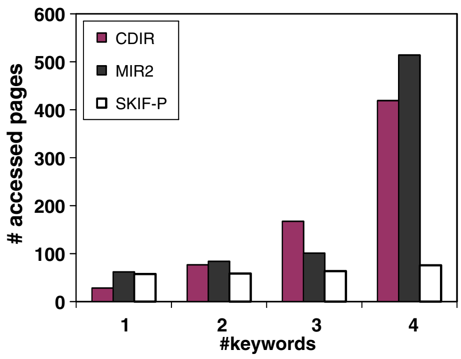
[12] $\beta = 0.1$ and the number of clusters $= 5$.

[13] Since the results for DATASET1 and DATASET2 were very similar, we only report the results of the larger dataset—DATASET2.

Fig. 8 Impact of $|K_q|$ on query cost



(a) Search Time



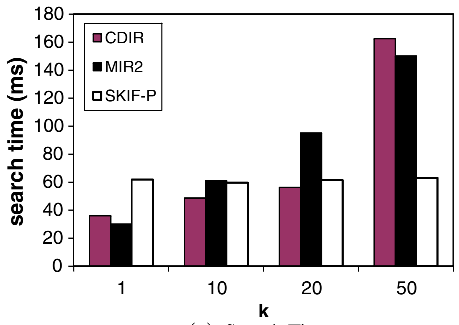
(b) I/O

to search larger and larger number of documents as $|K_q|$ increases. On the other hand, SKIF-P only searches for those documents containing the query keywords and therefore required to be scored. As a result, the SKIF-P outperforms the other two approaches even more significantly for a larger number of keywords.

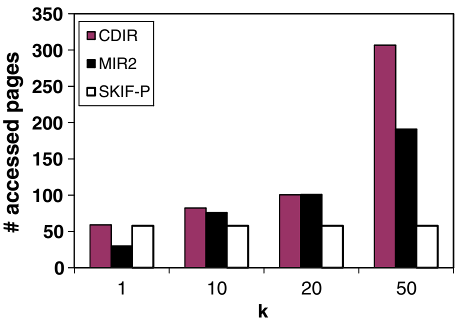
Number of requested results(k) With the second set of experiments, we evaluate the impact of the number of requested result k on the query performance. Again, we report the average query cost for each round. $|K_q|$ is fixed at two and α is fixed at 0.5 and k varies from 1 to 50. Figure 9a and b show the results for the search time and number of accessed pages, respectively. The first observation is that for SKIF-P, the query cost changes slightly as k increases. Since the average number of terms in the query as well as k are small, only few disk pages in the inverted lists of the few query terms are retrieved and processed. On the other hand, CDIR-Tree and MIR^2-tree perform worse as k grows since they have to access and process more entities in their corresponding trees. The second observation is that while all three approaches perform somehow similar for k from 1 to 10 (none is significantly superior to the others), SKIF-P significantly outperforms the other two approaches for larger values of k. The larger the value of k, the more significant is the difference between SKIF-P and the other two approaches.

Alpha (α) In the third set of experiments, we study the impact of the parameter α on the performance of SKIF-P and CDIR-tree (MIR^2-tree does not support

Fig. 9 Impact of k on query cost



(a) Search Time



(b) I/O

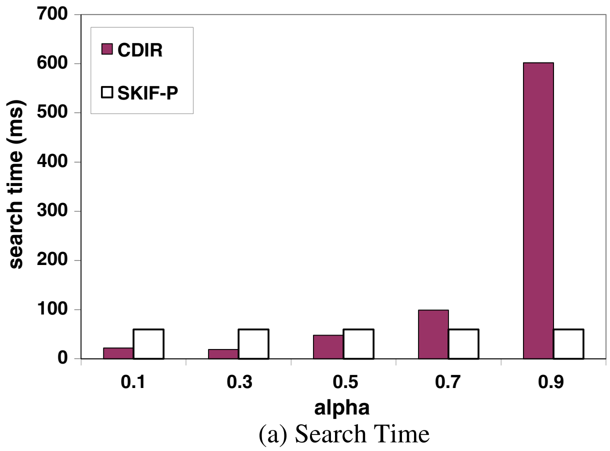
different weights for the textual and spatial relevances). As mentioned earlier, α is the parameter that assigns relative weights to the textual and spatial relevances. We fix $|K_q|$ at two and k at 10. Figure 10a and b show the results. The important observation is that the query cost for SKIF-P is weight-independent while CDIR-tree performs very poorly when the spatial relevance is more important (large α). Since CDIR-tree takes into account document similarity, it performs well when the textual relevance is given higher importance and performs poorly when the spatial relevance is given higher importance. On the contrary, SKIF-P performs well for all the cases, since the query processing is the same for both keywords and space and is not affected by the relative weights.

7.2 Performance: SKIF-P parameters

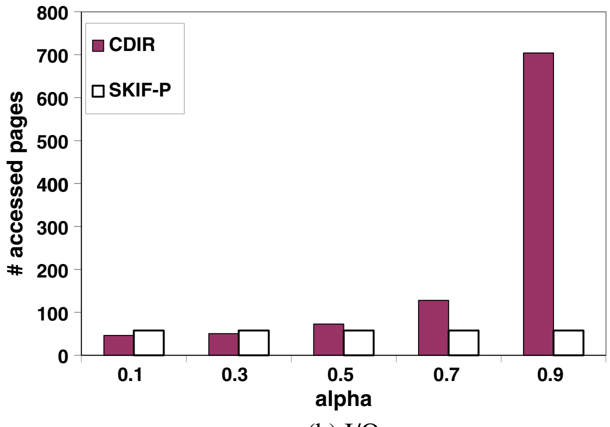
Next, we evaluate the performance of SKIF-P based on SKIF-P specific parameters: *number of cells* and the *threshold value* δ. We use DATASET1 for this set of experiments. Again, each query *round* consists of 100 queries. For each method, we report the average query cost in processing each round.

Number of cells In the first set of experiments, we study the impact of changing the *number of cells* on the performance of our system. In this set of experiments, we change the grid resolution from 2*2 cells to 400*400 cells and report the performance of SKIF-P. $|K_q|$ is fixed at 2 and α is fixed at 0.5, k is fixed at 10 and δ is equal to two cells. Figure 11a and b show the results for search time and number of pages accessed,

Fig. 10 Impact of α on query cost



(a) Search Time

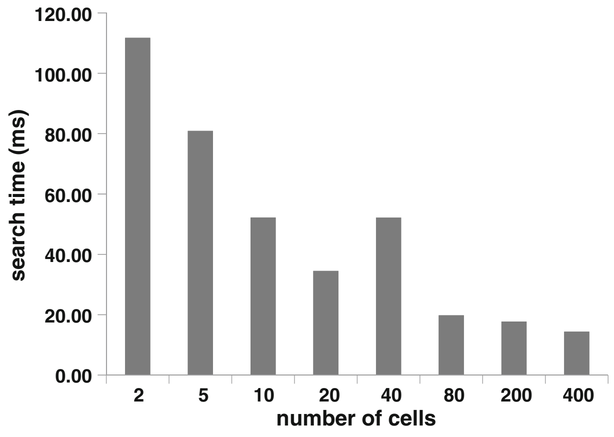


(b) I/O

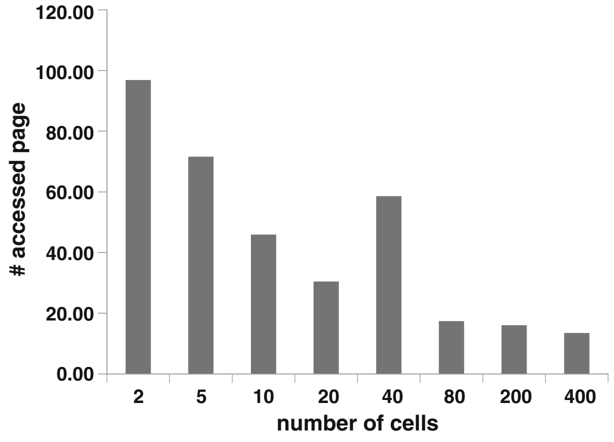
respectively. Both figures convey similar observations. The main observation is that most often, when number of cells increase (cell sizes decrease) the performance of the system improves. Smaller cells usually correspond to fewer documents in each cell and hence less number of entries in each cell's inverted list. This translates to fewer number of IOs and hence less processing time for a fixed δ (fixed number of cells are retrieved and processed for a fixed δ). In other words, larger cell sizes usually lead to larger search regions and hence more documents need to be searched. Since the number of cells is not the only factor affecting the performance, there may be cases that the above argument does not hold (e.g., 40×40 case in Fig. 11a and b). The query location, distribution of documents locations and also query's keywords and distribution of documents keywords may change this pattern.

Delta (δ) In the second set of experiments, we evaluate the impact of δ on the performance of SKIF-P. Again, $|K_q|$ is fixed at two and α is fixed at 0.5, k is fixed at 10 and number of cells is 40×40. The results are shown in Fig. 12a and b. In both figures, the main observation is that as expected both the number of IOs and also processing time increase as δ increases. Increasing δ results in retrieval and processing

Fig. 11 Impact of *number of cells* on query cost



(a) Search Time



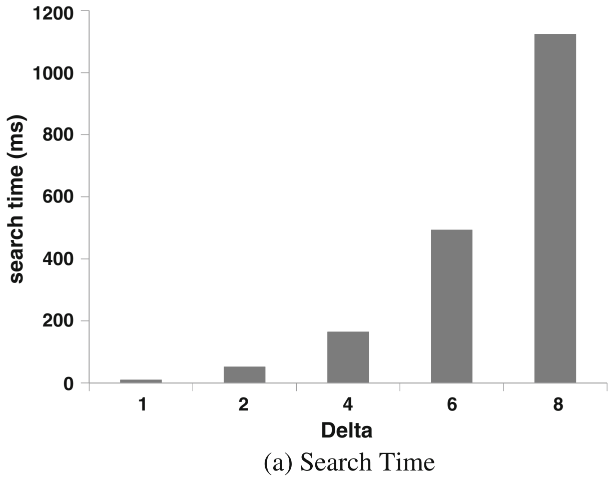
(b) I/O

of larger number of cells and hence larger number of disk IOs and subsequently larger amount of processing time.

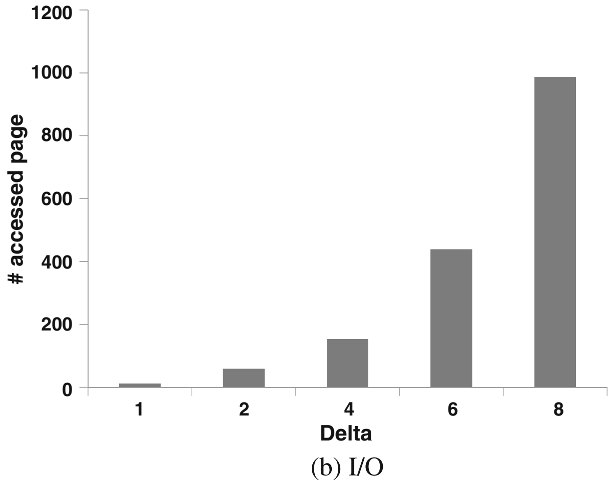
7.3 Accuracy

In this set of experiments, we evaluate the accuracy of our four proposed scoring approaches. We used the real dataset of DATASET2. Since spatial-keyword relevance ranking is new and no ground truth exists for our work, we conducted a user study to evaluate the effectiveness of our ranking methods. To conduct the user study, we utilized (and extended) the user study in [16] as our model and followed a similar procedure. We randomly selected 15 queries from our query set and found 10 volunteers. For each query, each volunteer was shown 6 result rankings, each one consisted of the top 10 results satisfying the query when the results were ranked with one of these approaches: *DSI*, *DSD*, *SSD*, *SSI*, *CDIR-tree* and *MIR*^{2}*-tree*. For *MIR*^{2}*-tree*, the relevance ranking is computed based on the distance of objects to the query point (the shorter the distance, the more relevant) for all objects that contain

Fig. 12 Impact of δ on query cost



(a) Search Time



(b) I/O

all the query keywords. For *CDIR-tree*, results are ranked according to a ranking function $f(D_\epsilon, P(q.keywords|d))$, where D_ϵ is the Euclidian distance between query q and document d's location and $P(q.keywords|d)$ is the probability of generating query keywords $(q.keywords)$ from the language models of the document d (for more details, refer to Section 2.1 of [10]). Each volunteer was asked to select all documents which were *relevant* to the query, in their opinion. They were not told how any of the rankings were produced. We used *R-precision* [17] to evaluate the results of various rankings. R-precision is defined as follows. Let a document be considered as *relevant* if at leat 6 of the 10 volunteers choose it as relevant for the query. Let *Rel* be a set that contains all such *relevant* documents and let $|Rel|$ be the size of that set. Then, the R-precision of each list is the fraction of the top $|Rel|$ documents that are deemed *relevant*. Hence, the higher the value of R-precision the more relevant the corresponding ranking. The R-precision of the six ranking approaches for each test query is shown in Table 2. We have also included the average R-precision for each ranking method. The first important observation is that for the majority of cases, our

Table 2 R-precision of various rankings

Query	DSI	DSD	SSD	SSI	MIR^2-tree	CDIR
1	0.97	0.97	0.97	0.97	0.00	0.4
2	1.00	1.00	0.83	0.83	0.00	0.6
3	1.00	1.00	1.00	1.00	1.00	1
4	1.00	1.00	0.90	0.90	1.00	1
5	1.00	1.00	0.90	0.90	1.00	0.9
6	0.90	0.78	0.84	0.84	0.00	0.2
7	0.94	0.88	0.94	0.88	0.88	0.2
8	0.83	0.83	0.83	0.83	0.83	0.83
9	1.00	1.00	1.00	1.00	1.00	1
10	1.00	1.00	1.00	1.00	1.00	1
11	1.00	1.00	1.00	1.00	1.00	1
12	1.00	1.00	1.00	0.94	0.84	0.67
13	0.97	0.94	0.94	0.94	0.00	0.67
14	1.00	1.00	1.00	1.00	1.00	1
15	1.00	1.00	1.00	0.94	1.00	1
Average	0.97	0.959	0.943	0.931	0.703	0.764

four proposed approaches generate results with R-precision equals to one, i.e., lists in which all the top |*Rel*| documents are *relevant*. The second observation is that the average R-precision for the rankings generated by our approaches is substantially higher than that of the other two rankings. For MIR^2-tree, when a query contains several keywords and no document in the collection includes all those keywords, the ranked result set will be empty. However, in all of our approaches, a document with only some of the query keywords (and having some spatial relevance to the query location) will be considered as a *relevant* document. As we expected, this is consistent with the evaluations received from the users. While CDIR-tree does not have such a problem, it still returns some documents in its top results which are considered not *relevant* by the majority of the users. For instance, there were cases in our user study that documents containing none of the query keywords were among the top

Table 3 Ranking preferred by users

Query	Preferred by majority
1	DSI
2	DSI
3	DSI
4	DSI
5	DSI
6	DSD
7	DSI
8	SSD
9	SSD
10	DSD
11	SSD
12	DSI
13	DSI
14	MIR^2-tree
15	DSI

results while the majority of the users felt differently. In contrary to CDIR-tree, we consider a document *relevant* if it is both *spatially relevant* and *textually relevant* (see Section 3.1). This view is consistent with the result of the user study. Finally, Table 3 shows the rankings (actual order) preferred by the majority of the users. For nearly all the queries, a majority of the users preferred one of our proposed scoring methods. These results, further confirms the effectiveness of our proposed approaches.

7.4 Accuracy: SKIF-P parameters

In this section, we evaluate the impact of the threshold value δ and also the type of the decay function on the accuracy of our system. We also use two new (and standard) metrics to evaluate the accuracy: Precision at k and nDCG at k. We used the real-world Flickr dataset (DATASET3). As explained earlier, this dataset contains information about geo-tagged images on Flickr website. We started with a set of 30,000 random images from Flickr. In the processing of this data, we removed the images with no location and also removed any image outside California. After these steps, the final dataset size was reduced to 8,964 documents (images). We used *SSI* as our scoring mechanism (the other three have slightly better but almost identical accuracy, see Table 2), and again we used a 40×40 grid of cells. Values of $gamma$ and $lambda$ (for polynomial and exponentially decays, respectively) are set to 1.8.

Approaches We ran our experiments for three decay functions discussed in Section 4 and for three ranges of δ: *short* (0–1 cells), *medium* (2–5 cells) and *large* (6–10 cells). The combination of the three values of δ and the three spatial decay functions generates a total of 9 approaches.

Queries We generated a set of 15 queries from different keywords in DATASET3 and randomly assigned them a two-dimensional point in California.

Relevance assessment After computing top-5 results for each of our 15 queries using all the 9 approaches, we ran a user study using Amazon Mechanical Turk.[14] One task (hit) was generated for each query. Each query was run using all 9 approaches and top-5 results were returned for each approach. Then, all the results from all the methods were combined together. A Google map mashup web-page with markers representing these results (Flickr images) were generated. Also, query location (as a separate marker on the map) alongside the query keywords were provided to the workers. Workers could click on each marker and see the keywords associated with that marker (image) as well. Workers could choose whether the result (marker plus the keywords) is *relevant* or *non-relevant*. Workers could also add their comments/explaination for each assessment. Each task (query assessment) was assessed by ten workers. We used workers with HIT approval rate greater than or equal to 90 (meaning that more than 90% of their past assessments were correct). Each worker was rewarded \$0.04 by completion of each assessment. Overall, workers chose *relevant* for 73% of the assessments and *non-relevant* for 27% of the assessments.

[14]https://www.mturk.com/

Table 4 Precision@k of various rankings

		Decay function			Average
		Exponential	Polynomial	Windows	
Delta (δ)	Short	0.613	0.626	0.626	0.622
	Medium	0.76	0.746	0.76	0.755
	Long	0.733	0.746	0.68	0.72
	Average	0.702	0.706	0.688	

Metrics We evaluated the accuracy of the methods under comparison using two standard metrics: precision at k and nDCG at k. In calculating precision at k, we consider a document *relevant* if a majority of the workers assessed that document as relevant and *non-relevant* otherwise. When computing nDCG at k, we consider the average relevance given by the users to each document, interpreting *relevant* as score 1 and *non-relevant* as 0, respectively. The ideal ranking is calculated based on these average relevance scores.

Results The results of our relevance assessments with $k = 5$ and using the nine approaches using precision at k and nDCG at k are shown in Tables 4 and 5, respectively. For the precision@5, the first observation is that all of the evaluated methods generate accurate results (precision@5 larger than 0.6). The second observation is that, the approaches with medium and large values of δ perform the search more accurately. Smaller values of δ may lead to smaller number of results in the final result set and hence smaller number of relevant results. On the other hand, larger values of δ usually leads to more relevant results in the final result set. As you can see in Table 4, the medium values of δ are slightly better than the large values of δ in our experiments. This is because, for some cases (especially for windows decay function) when value of δ is large and there are many relevant documents to the query, some documents farther to the query location become relevant and end up in the final result set. Some users do not consider these document relevant while others seem to evaluate them as relevant. The final observation is that there is no one superior decay type with regards to the different decay functions. All three generate fairly similar and accurate results while the windows decay function generates a slightly less accurate result set.

As for the nDCG@5, the first observation is that all the evaluated methods rank the results very accurately (nDCG@5 larger than 0.8). In other words, the average relevance values of the top-5 results (with their respected ranks) is very similar to the relevance values of the best possible ranking. The second observation is that as expected rankings improve when δ increases. Using smaller values of δ may lead

Table 5 nDCG@k of various rankings

		Decay function			Average
		Exponential	Polynomial	Windows	
Delta (δ)	Short	0.812	0.811	0.812	0.812
	Medium	0.913	0.917	0.913	0.915
	Long	0.934	0.982	0.934	0.950
	Average	0.886	0.904	0.886	

to missing some relevant results in the final result set (and ranking). On the other hand, larger values of δ usually lessens the probability of missing relevant results and hence the final ranking can be done on a larger set of relevant documents. Hence, generating a more accurate (and complete) ranking. The last observation is that although there is still no superior decay function among the decay functions, the polynomial type generates more accurate rankings. Based on the users' input, it seems that the exponential decay is a little too fast of a decay for our setting and users prefer the polynomial decay slightly more. As a conclusion, all the 9 approaches generate accurate results and very accurate rankings while the accuracy improves for larger δ values. All the other properties seem to be very similar (although slightly different from case to case).

8 Conclusions

In this paper we introduced the problem of ranking the spatial and textual features of web documents. We proposed new scoring methods to rank documents by seamlessly combining their spatial and textual features. We also proposed an efficient point-based index structure which handles the spatial and textual features of data simultaneously and also supports the spatial-keyword relevance ranking. In particular we introduced SKIF-P and showed how it is used to search and rank the documents efficiently. We experimentally studied our methods, which proved its superior performance and accuracy.

Acknowledgements Ali Khodaei and Cyrus Shahabi's research has been funded in part by NSF grants CNS-0831505 (CyberTrust) and IS-1115153, the USC Integrated Media Systems Center (IMSC), and unrestricted cash and equipment gifts from Google, Microsoft and Qualcomm. Chen Li is partially supported by the US NSF IIS 1030002 award and the National Natural Science Foundation of China (No. 61129002). Any opinions,findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Khodaei A, Shahabi C, Li C (2010) Hybrid indexing and seamless ranking of spatial and textual features of web documents. In: DEXA, pp 450–466
2. Zhou Y, Xie X, Wang C, Gong Y, Ma W-Y (2005) Hybrid index structures for location-based web search. In: CIKM, pp 155–162
3. Hariharan R, Hore B, Li C, Mehrotra S (2007) Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. In: SSDBM, p 16
4. De Felipe I, Hristidis V, Rishe N (2008) Keyword search on spatial databases. In: ICDE
5. Zobel J, Moffat A (2006) Inverted files for text search engines. ACM Comput Surv 38(2):6
6. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley, Reading
7. Chen Y, Suel T, Markowetz A (2006) Efficient query processing in geographic web search engines. In: SIGMOD, pp 277–288
8. McCurley KS (2001) Geospatial mapping and navigation of the web. In: WWW, pp 221–229
9. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Readings in information retrieval. Morgan Kaufmann Publishers Inc
10. Cong G, Jensen CS, Wu D (2009) Efficient retrieval of the top-k most relevant spatial web objects. In: Proc. VLDB endow. 2, 1 (August 2009), pp 337–348
11. Vaid S, Jones CB, Joho H, Sanderson M (2005) Spatio-textual indexing for geographical search on the web. In: SSTD

12. Amitay E, HarEl N, Sivan R, Soffer A (2004) Web-a-where: geotagging web content. In: SIGIR, pp 273–280
13. Ding J, Gravano L, Shivakumar N (2000) Computing geographical scopes of web resources. In: VLDB, pp 545–556
14. Gao W, Lee HC, Miao Y (2006) Geographically focused collaborative crawling. In: WWW
15. Zobel J (1995) Adding compression to a full-text retrieval system. Softw Pract Exp 25(8):891–903
16. Haveliwala T (2002) Topic-sensitive PageRank. In: WWW
17. Manning C, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
18. Jeong B-S, Omiecinski E (1995) Inverted file partitioning schemes in multiple disk systems. IEEE Trans Parallel Distrib Syst 6:2
19. Alsubaiee S, Behm A, Chen L (2010) Supporting location-based approximate-keyword queries. In: Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems (GIS '10), pp 61–70
20. Wang Z, Du M, Le J (2009) gR*-tree: an index for querying approximate keywords in geographic information system. In: Information engineering and computer science
21. Zhang D, Chee YM, Mondal A, Tung AKH, Kitsuregawa M (2009) Keyword search in spatial databases: towards searching by document. In: ICDE, pp 688–699
22. Cormode R, Shkapenyuk V, Srivastava D, Xu B (2009) Forward decay: a practical time decay model for streaming systems. In: ICDE
23. Cohen E, Strauss MJ (2006) Maintaining time-decaying stream aggregates. J Algorithms 59:1
24. Cao X, Cong G, Jensen CS (2010) Retrieving top-k prestige-based relevant spatial web objects. In: Proc. VLDB endow., vol 3, pp 1–2
25. Tobler W (1970) A computer movie simulating urban growth in the Detroit region. Econ Geogr 46:234–240
26. Long X, Suel T (2005) Three-level caching for effcient query processing in large web search engines. In: WWW, pp 257–266

Ali Khodaei received his B.S. degree in computer engineering from Iran National University (Shahid Beheshti University), Tehran, Iran, in 2003 and his M.S. degree in information and computer science from University of California, Irvine, in 2006. He has worked (and interned) as a researcher/developer/manager in several companies and research institutes including Microsoft, Samsung R&D center (SISA), VESystems and Integrated Media Systems Center (IMSC). He is currently working towards his Ph.D. degree in computer science at the University of Southern California where he is a research assistant working on web search and information retrieval, geo-spatial databases and location-based services at Information Laboratory (InfoLab) at the Computer Science Department of the University of Southern California.

Cyrus Shahabi is a Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department and also the Director of the NSF's Integrated Media Systems Center (IMSC) at the University of Southern California. He is also the CTO and co-founder of a USC spin-off, Geosemble Technologies. He received his B.S. in Computer Engineering from Sharif University of Technology in 1989 and then his M.S. and Ph.D. Degrees in Computer Science from the University of Southern California in May 1993 and August 1996, respectively. He authored two books and more than hundred-fifty research papers in the areas of databases, GIS and multimedia. Dr. Shahabi has received funding from several agencies such as NSF, NASA, NIH, DARPA, AFRL, and DHS as well as several industries such as Google, Microsoft, NCR and Chevron. He was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems (TPDS) from 2004 to 2009. He is currently on the editorial board of the VLDB Journal, IEEE Transactions on Knowledge and Data Engineering (TKDE), ACM Computers in Entertainment and Journal of Spatial Information Science. He is the founding chair of IEEE NetDB workshop and also the general co-chair of ACM GIS 2007, 2008 and 2009. He chaired the nomination committee of ACM SIGSPATIAL for the 2011–2014 terms. He regularly serves on the program committee of major conferences such as VLDB, ACM SIGMOD, IEEE ICDE, ACM SIGKDD, and ACM Multimedia. Dr. Shahabi is a recipient of the ACM Distinguished Scientist award in 2009, the 2003 U.S. Presidential Early Career Awards for Scientists and Engineers (PECASE), the NSF CAREER award in 2002, and the 2001 Okawa Foundation Research Grant for Information and Telecommunications. He was an organizer of the 2011 National Academy of Engineering Japan-America Frontiers of Engineering program, an invited speaker in the 2010 National Research Council (of the National Academies) Committee on New Research Directions for the National Geospatial-Intelligence Agency, and a participant in the 2005 National Academy of Engineering "Frontiers of Engineering" program.

Chen Li is an associate professor in the Department of Computer Science at the University of California, Irvine. He received his Ph.D. degree in Computer Science from Stanford University in

2001, and his M.S. and B.S. in Computer Science from Tsinghua University, China, in 1996 and 1994, respectively. He received a National Science Foundation CAREER Award in 2003 and a few other NSF grants and industry gifts. He was once a part-time Visiting Research Scientist at Google. His research interests are in the fields of data management and information search, including text search, data cleansing, and data integration. He is a founder of BiMaple.com.