Hybrid Indexing and Seamless Ranking of Spatial and Textual Features of Web Documents

Ali Khodaei[1], Cyrus Shahabi[1], and Chen Li[2]

[1] Department of Computer Science, University of Southern California, Los Angeles,CA, USA
[2] Department of Computer Science, University of California at Irvine, CA, USA
{khodaei,shahabi}@usc.edu, chenli@ics.uci.edu

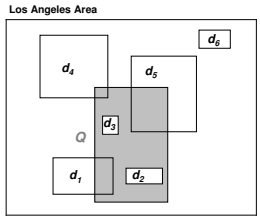
Abstract. There is a significant commercial and research interest in location-based web search engines. Given a number of search keywords and one or more locations that a user is interested in, a location-based web search retrieves and ranks the most textually and spatially relevant web pages. In this 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. 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 to handle location-based web searches 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.

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 schools at certain regions, 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 pages, and each page describes objects for a specific location, such as a district, a city, or a county. Objects can be schools, real-state agencies, golf courses, and sports teams. We want to build a system to allow users to search on these documents. Consider a user, Mike, who moves to the central part of Los Angeles. He likes to play soccer, and wants to find soccer leagues in this area so that he can choose one to join. He submits a query to the system with two keywords "`soccer league`" and specifies "`Central Los Angeles`" as the location restriction. Figure 1 shows the location of his query represented as a shaded region. Our goal is to find the best documents that are of interest to Mike.

Suppose there are six documents in the repository with locations close to the Central Los Angeles. Figure 1(a) shows these locations represented as rectangles. In addition,



(a)

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

(b)

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

each document has text keywords in its content. Figure 1(b) shows the frequencies of the two 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.

One way to answer the query is to find the documents with a location contained in the query region and with the two query keywords, as suggested in several studies in the literature [2, 6, 10]. Using this approach, we can only find document d_3 as an answer, since it satisfies both conditions. Document d_5 is not an answer since even though it has both query keywords, its region is not totally contained in the query region. One major limitation of this approach is that many documents with partial spatial and/or textual matching will not be considered, even though they include information that could be interesting to the user.

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 region mentioned in the query. Document d_6 is not very relevant to the query, since its region is far from the query region. The other five documents are all overlapping with the query region, and thus could be potentially of interest to the user. Hence, we need to rank them since the user may be only interested in the most relevant documents. However, it is not clear how to measure the relevance of the documents to user query. For example, it is clear that document d_3 should have a high relevance since its region is contained in the query region, and both query keywords appear in the document. However, it is not clear how relevant d_2 is to the query, since even though its region is contained in the query region, it does not have the keyword "`league`." The other documents, d_1, d_4, and d_5, all have the two query keywords, but with different frequencies, and they have different amounts of overlapping areas with the query region.

In this paper, we present a ranking method that considers both the spatial overlap of a document with a query and the frequencies of the query keywords in 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 good ranking method for documents, a natural question to ask is how to efficiently index and search the location-specific documents. There are several chal-

lenges. 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), making 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 index structure into existing search engines.

To solve the above problems, we propose a new hybrid index structure called *Spatial-Keyword Inverted File* ("SKIF" for short), which can handle the spatial and textual features of data simultaneously and in a similar manner. SKIF is an inverted file capable of indexing and searching on both textual and spatial data in a similar, integrated manner. Towards this end, 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, and present two efficient algorithms for answering a ranking query using SKIF.

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 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 an experimental evaluation on real and synthetic datasets to show that our techniques can answer ranking queries efficiently and accurately.

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.

Hybrid index structures combine the textual and spatial indexes into one index structure. Two basic designs introduced in [1] are *inverted file-R*-tree* and *R*-tree-inverted file*. *Inverted file-R*-tree* is essentially an inverted file on top of R*-trees. 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. *R*-tree-inverted file* is an R*-tree on top of inverted files. 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 [6] and [10].

Both index structures use grid as a spatial index and inverted file as a textual index. With both approaches, spatial and textual search processes are separated and query processing require two stages. Neither of the approaches can support spatial- keyword relevance ranking.

Several improved hybrid index structures are introduced more recently. In [2] 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. Another hybrid index structure called IR^2-tree is presented in [3], 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. 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 [1] and [2] 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.

Very recently another hybrid index structure called IR-tree is presented, which also combines R-tree with the inverted files [9]. 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.

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, 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. 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 [9] is an accurate spatial-keyword relevance ranking (see Section 6).

There are many other relevant topics such as extraction of geographical information [7, 12], geo-coding of documents' locations [11], and geographic crawling[13]. 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 rep-

resented by a minimum bounding rectangle (MBR) although any other arbitrary shape can be used.

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 query specified as one or more minimum bounding rectangles and K_q is a set of keywords in the query.

Spatial relevance: Spatial relevance between a document d and the query q is defined based on the type of the spatial relationship that exists between L_d and L_q. We focus only on the *overlap* relationship, although our approach can easily be extended to cover other spatial relationships. Subsequently, we define spatial relevance as follows: A document d and the query q are spatially relevant if at least one of the query's MBRs has a non-empty intersection with one of the document's MBRs, i.e., $L_q \cap L_d \neq \emptyset$. The larger the area of the intersection 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 $kRel_q(d)$.

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: $F_q(d) = \alpha.sRel_q(d) + (1 - \alpha).kRel_q(d)$. 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 4 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

tf-idf Score All current 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 [4]: (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 [5, 8]. 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 Equation 1.

$$w_{q,k} = \ln(1 + \frac{n}{f_k}); \quad w_{d,k} = \ln(1 + \overline{f_{d,k}});$$
$$W_d = \sqrt{\sum_k w_{d,k}^2}; \quad W_q = \sqrt{\sum_k w_{q,k}^2}; \quad S_{q,d} = \frac{\sum_k w_{d,k}.w_{q,k}}{W_d.W_q}. \tag{1}$$

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 $tRel_q(d)$.

keyword k	f_k	Inverted list for k
soccer	5	$\langle 1,1\rangle\langle 2,1\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle$
league	5	$\langle 1,0.8\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle\langle 6,1\rangle$

Fig. 2. Inverted file for Example 1.

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 [4]. 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 [4]. A complete inverted file for Example 1 is shown in Figure 2.

4 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.

4.1 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 set of cell identifiers. Since we are using overlap as our main spatial query type, these cells are defined as the cells which overlap with the document location. With spatial tf-idf, the overlap of a cell with the document is analogous to the existence of a keyword in document with tf-idf. However, knowing the overlapping cells is not enough. We need to know how well a cell describes the spatial content of the document. We use the overlap area between each cell and the document to provide a measure of how well that cell describes the document. Analogous to *frequency of term t in document d*, we define *frequency of cell c in document d* as follows: $f_{d,c} = \frac{L_d \cap c}{c}$ which is the area of overlap between the document location L_d and cell c divided by the area of cell c. 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 more the overlap, the better this cell describes the document location and viceversa.
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 overlap largely with a document; 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. 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 having a large overlap area with the document. The larger the overlap, the better that cell describes the document location. For example, in Figure 3, cell c_8 better describes the document d_2 than cell c_9. Property (3) states that less wight should be given to those documents whose locations cover more cells. 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. To illustrate, consider Figure 3, where both documents $d1$ and $d2$ contain the query location and its corresponding cell (i.e. c_5). In other words, they have equal spatial tf scores for cell c_5 . Cell c_5 also have identical spatial idf for all documents. Under these conditions (equal tf scores and equal idf scores), the smaller document (d_1) should be ranked higher. 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, the idf

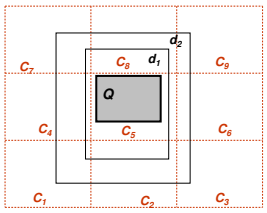


Fig. 3. Properties (2) and (3)

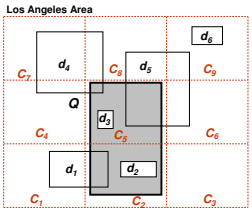


Fig. 4. Example 1 on the grid

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 overlapping with many documents - is a very meaningful cell for some users/applications, while for some others, a distinctive cell (location) - cell appearing in few documents - is more meaningful. (in Example 1, one user may look for more popular locations for soccer leagues 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).

4.2 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.

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: $T_{d_1} = \{\texttt{soccer}, \texttt{league}, \texttt{c}_1, \texttt{c}_2\}$ (see Figures 1(b) and 4). In order to be able to define a single similarity measure capturing both the textual and spatial relevances, we define the following parameters:

- $f_{d,t}$: the 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 Equation 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.

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

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.

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 very large document location 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:

$$w_{q,c} = \begin{cases} \ln(1+\frac{n}{f_c}) & \text{if inverted document frequency} \\ \ln(1+\frac{f_c}{n}) & \text{if direct document frequency} \end{cases}; \quad w_{d,c} = \ln(1+\overline{f_{d,c}});$$
$$W'_d = \sqrt{\sum_c w_{d,c}^2}; \qquad W'_q = \sqrt{\sum_c w_{q,c}^2}; \qquad S'_{q,d} = \frac{\sum_c w_{d,c}.w_{q,c}}{W'_d.W'_q}. \qquad (3)$$

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 Equation 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 $w_{q,c} = \ln(1+\frac{n}{f_c})$
2. Single-Score with Direct document frequency (SSD)
   Where $skRel_q(d) = \widehat{S_{q,d}}$ and $w_{q,c} = \ln(1+\frac{f_c}{n})$

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

5 Spatial-Keyword Inverted File

Spatial-Keyword Inverted File (SKIF) 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 and the information it stores. Next, we show how spatial-keyword query evaluation is performed using SKIF. Two algorithms corresponding to our two approaches are presented. Finally, we discuss briefly how SKIF can be extended to more general cases.

5.1 SKIF Structure

Since SKIF is an inverted file, its structure is very similar to the structure of the regular inverted files. SKIF 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 term which is used to help calculate the tf and idf scores. The second component of SKIF 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 4 redraws the Example 1 on the grid and Figure 5 shows the complete SKIF for Example 1.

term t	f_t	type	Spatial-Keyword Inverted List for t
`soccer`	5	1	$\langle 1,1\rangle\langle 2,1\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle$
`league`	5	1	$\langle 1,0.8\rangle\langle 3,1\rangle\langle 4,1\rangle\langle 5,1\rangle\langle 6,1\rangle$
c_1	1	0	$\langle 1,1\rangle$
c_2	2	0	$\langle 1,0.55\rangle\langle 2,1\rangle$
c_4	1	0	$\langle 4,0.25\rangle$
c_5	3	0	$\langle 3,1\rangle\langle 4,0.06\rangle\langle 5,1\rangle$
c_6	1	0	$\langle 5,0.35\rangle$
c_7	1	0	$\langle 4,1\rangle$
c_8	2	0	$\langle 4,0.3\rangle\langle 5,0.65\rangle$
c_9	2	0	$\langle 5,0.25\rangle\langle 6,1\rangle$

Fig. 5. Spatial-keyword inverted file for Example 1.

5.2 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 overlapping with the document location L_q . After calculating C_q, we define the set of terms associated with each query by T_q as follows: $T_q = K_q \cup C_q$.

Figures 1 and 2 show the algorithms to perform top-k spatial-keyword search using SKIF 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.

Algorithm 1 single-score approach

Allocate an accumulator A_d for each document d
Set $A_d \leftarrow 0$
for each query term t in q **do**
    Calculate $w_{q,t}$ and fetch the inverted list for t
    for each pair $< d, f_{d,t} >$ in the inverted list **do**
        Calculate $w_{d,t}$
        Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
Read the array of $\widehat{W_d}$ values
for each $A_d > 0$ **do**
    Set $\widehat{S_d} \leftarrow A_d \div \widehat{W_d}$
Identify the k greatest $\widehat{S_d}$ values

Algorithm 2 double-score approach

Allocate two accumulators A_d and A'_d for each document d
Set $A_d \leftarrow 0$
Set $A'_d \leftarrow 0$
for each query term t in q **do**
    Calculate $w_{q,t}$ and fetch the inverted list for t
    for each pair $< d, f_{d,t} >$ in the inverted list **do**
        Calculate $w_{d,t}$
        if type of t is a keyword **then**
            Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
        else
            Set $A'_d \leftarrow A'_d + w_{q,t} \times w_{d,t}$
Read the array of W_d values
for each $A_d > 0$ **do**
    Set $S_d \leftarrow A_d \div W_d$
Read the array of W'_d values
for each $A'_d > 0$ **do**
    Set $S'_d \leftarrow A'_d \div W'_d$
    if $A_d > 0$ **then**
        $\widehat{S_d} = \alpha.S'_d + (1 - \alpha).S_d$
Identify the k greatest $\widehat{S_d}$ values

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 . Finally, the k largest documents are identified and will be returned to the user.

5.3 Generalization

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

Multiple Locations: One of the advantages of our technique is that there is no limiting constraint on the representation of the document location. Instead of treating the document location as one large and sparse MBR, we can use several sperate, disjoint locations using SKIF. This is feasible because our final spatial relevance score can be computed by separately computing the spatial score of each cell intersecting with the various document locations. Another advantage of SKIF is its capability to represent the document location as any arbitrary shape and not necessarily as MBR or rectangle. The only information we need to calculate for spatial tf-idf score is the area of overlap between each cell and each document location.

Points: We assumed that each document location is a region. In the context of the web, this is a reasonable assumption, still in the rare cases when the document location is one geographical point (point p) we can generalize our approach as follows. A circle centered at p with radius r is constructed. The MBR covering the circle is our new document location. Radius r is a parameter determined by user and also the context of the web-page (if available).

Weights: When querying the system, there are two types of weight users may want to manipulate 1) setting different weights to the spatial and textual relevance, and 2) setting different weights to different terms in the query. For the first scenario, we have used the parameter α in this paper. SKIF can also support setting 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 a wide possibility of complicated queries to the users.

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 is very similar to the structure of the regular inverted files, same techniques used in regular search engines (built on inverted files) can be applied for our location-based search engine (built on SKIF).

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 using a body of work exists in this field. For example, *compression* techniques are very popular for inverted files [4, 14]. Since the structure of the inverted lists are identical with both SKIF and regular inverted files, no change is needed to apply the same compression techniques on SKIF. More interestingly, some of the optimization techniques seem to work better on SKIF. For instance, *caching* is another technique used in existing search engines. It is easy to see that with SKIF, 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.

6 Experiments

In this section, we experimentally evaluate the performance and accuracy of SKIF. Comparison is done with the most efficient proposed solutions: *MIR^2-tree* [3] and *CDIR-tree* [9] which are optimized versions of IR^2-tree and IR-tree, respectively. Since both of *MIR^2-tree* and *CDIR-tree* use query points instead of query regions, we apply the following adjustment: Each query is executed using *MIR^2-tree* and *CDIR-tree* separately for random query point q and for total number of results k. Subsequently, the farthest document in the union of the result sets is identified. Let r be the distance between q and this farthest document. We construct a circle centered at q with radius r, the MBR covering the circle is considered as the query location (L_q).

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

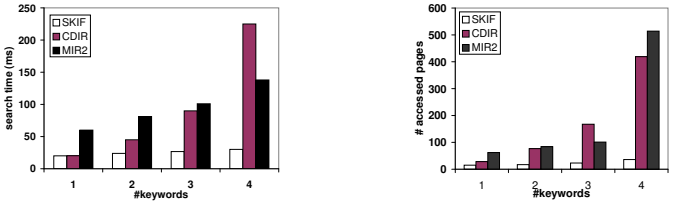
Table 1. Dataset Details

Our experiments use two datasets which their 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 regions using Yahoo! Placemaker (http://developer.yahoo.com/geo/placemaker/). We run our experiments on all the documents in US. For DATASET1, we partition the space into $100km \times 100km$

cells. DATASET2 is generated synthetically. A set of keywords (from 1 to 500) and one location are assigned randomly to each document. Space is partitioned into 225×225 cells. The documents' keywords and locations are uniformly distributed.

Each query contains 1 to 4 randomly generated keywords and one rectangle. 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 [3] and [9], respectively (e.g. signature length = 189 bytes, $\beta = 0.1$, number of clusters = 5, etc.) . Experiments were run on a machine with an Intel Core2 Duo 3.16 GHz CPU and with 4GB main memory. Due to space limitation and since the results for DATASET1 and DATASET2 were very similar, we only report the results for DATASET2.

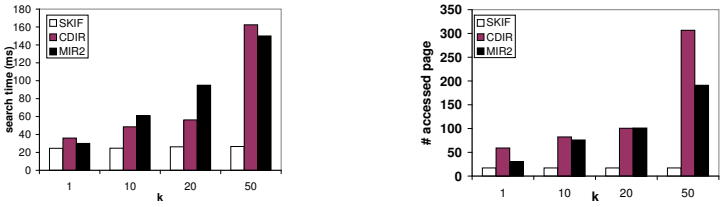
Performance. With the first set of experiments, we evaluate the impact of the number of keywords in each query $|K_q|$ on query cost. In this set of experiments, we vary $|K_q|$ from 1 to 4 while fixing k at 10 and α at 0.5. For each method, we report the average query cost in processing each round. The results are shown in Figures 6(a) and 6(b). For almost all the cases, SKIF significantly 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 is very marginal. While the I/O costs of CDIR-tree and MIR^2-tre increase by a factor of 15 and 8 respectively, SKIF's query cost barely doubles when the number of keywords grows from 1 to 4. Both CDIR-tree and MIR^2-tree will 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 with each node of CDIR-tree is very similar, which makes the textual relevance pruning less effective. Therefore, both approaches need to search larger and larger number of documents as $|K_q|$ increases. On the other hand, SKIF only searches for those documents containing the query keywords and therefore required to be scored.



(a) Search Time (b) I/O

Fig. 6. Impact of $|K_q|$ on query cost

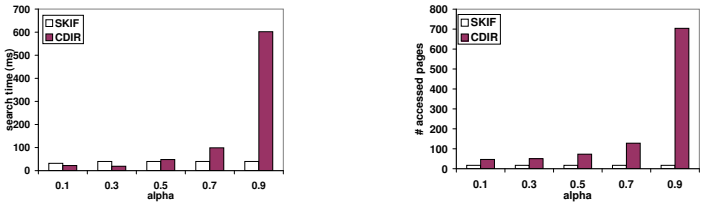
With the second set of experiments, we evaluate the impact of the number of requested result k on the query performance. Again, we report the average query cost for each round. $|K_q|$ is fixed at two and α is fixed at 0.5 and k varies from 1 to 50. Figures 7(a) and 7(b) show the results for search time and number of page accesses, respectively. The first observation is that for SKIF, 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, CIDR-Tree and MIR^2-tree perform worse as k grows since they have to access and process more entities in their corresponding trees.



(a) Search Time (b) I/O

Fig. 7. Impact of k on query cost

In the third set of experiments, we study the impact of the parameter α on the performance of SKIF and CDIR-tree. 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. Figures 8(a) and 8(b) show the results. The important observation is that the query cost for SKIF 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 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.



(a) Search Time (b) I/O

Fig. 8. Impact of α on query cost

Accuracy. Our final set of experiments was conducted to evaluate the accuracy of our four proposed scoring approaches. 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 the user study in [15] (well-known paper in information retrieval) as our model and followed a similar procedure. We randomly selected 10 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*. 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* [16] to evaluate the result s 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 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.

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.

Query	DSI	DSD	SSD	SSI	MIR^2-tree	CDIR
1	1	1	1	1	1.00	0.67
2	1	1	1	1	1.00	1.00
3	1	1	1	0.95	1.00	1.00
4	1	1	0.8	0.8	0.00	0.10
5	0.9	0.7	0.9	0.8	0.00	0.20
6	1	1	1	1	0.88	0.88
7	1	1	1	1	0.00	0.38
8	1	1	1	1	1.00	1.00
9	1	1	1	1	1.00	0.80
10	1	1	1	1	1.00	1.00
Average	0.99	0.97	0.97	0.96	0.69	0.70

Table 2. R-precision of various rankings

Query	Preferred by Majority
1	DSI
2	DSI
3	DSI
4	DSD
5	DSI
6	DSD
7	SSD
8	MIR^2-tree
9	MIR^2-tree
10	DSI

Table 3. Ranking preferred by users

7 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 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 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.

Acknowledgments. This research has been funded in part by NSF grant CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. 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. Zhou, Y. *et al* , Hybrid index structures for location-based web search, CIKM, 2005.
2. Hariharan, R. *et al* , Processing spatial-Keyword (SK) queries in geographic information retrieval (GIR) systems, SSDBM, 2007.
3. De Felipe, I. *et al* , Keyword search on spatial databases, ICDE, 2008.
4. Zobel, J. *et al* , Inverted files for text search engines, ACM Comput., 2006.
5. Baeza-Yates, R. *et al* , Modern information retrieval, Addison Wesley, 1999.
6. Chen, Y., Efficient query processing in geographic web search engines, SIGMOD, 2006.
7. McCurley, K.S. *et al* , Geospatial mapping and Navigation of the Web, WWW, 2001.
8. Salton, G. *et al* , Term-Weighting approaches in automatic text retrieval, 1988.
9. Cong, G., Efficient retrieval of the top-k most relevant spatial web objects, PVLDB, 2009.
10. Vaid, S., *et al* , Spatio-textual indexing for geographical search on the web, SSTD, 2005.
11. Amitay, E., *et al* , Web-a-where: geotagging web content, SIGIR, 2004.
12. Ding, J., *et al* , Computing geographical scopes of web resources, VLDB, 2000.
13. Gao, W., *et al* , Geographically focused collaborative crawling, WWW, 2006.
14. Zobel, J., *et al* , Adding compression to a full-text retrieval system, Sof. Prac. Exp., 1995.
15. Haveliwala, T., *et al* , Topic-sensitive PageRank, www, 2002.
16. Manning, C., *et al* , Introduction to information retrieval, Cambridge university press, 2008.