

Preference-based Top- k Spatial Keyword Queries *

Jinzeng Zhang
Information School
Renmin University of China
Beijing, China
zajize@ruc.edu.cn

Dongqi Liu
Information School
Renmin University of China
Beijing, China
liudongqi@ruc.edu.cn

Xiaofeng Meng
Information School
Renmin University of China
Beijing, China
xfmeng@ruc.edu.cn

ABSTRACT

With proliferation of geo-positioning and geo-tagging, spatial keyword query has been an attractive and challenging topic that blooms various interesting applications in spatial databases. However, spatial keyword queries in those fields do not always satisfy users requirements. One reason for those queries contain fuzzy linguistic labels that often embody users preference, e.g., finding hotels near Zhongguancun with *moderate* price. In this paper, we define Preference-based Top- k Spatial Keyword (PTkSK) query for coping with fuzzy labels. Existing works only consider on spatial locations, textual descriptions and overlook users preference. In order to answer PTkSK queries efficiently, we propose a hybrid index structure called AIR-tree (Attribute-Inverted File R-tree), which combines location proximity with preference similarity and textual relevance. Given a query with fuzzy labels, we propose an efficient strategy of computing users preference to transform these labels into the $[0,1]$ interval, and design two effective search methods to find relevant objects for satisfying users requirements. Our experiments on synthetic and real datasets demonstrate that our proposed methods are scalable and efficient.

ACM Classification Keywords

H.2.8 Database Management: Database Applications spatial database and GIS

General Terms

Algorithms, Experimentation, Performance

Author Keywords

Spatial Keyword Queries, Preference, Mapping Function

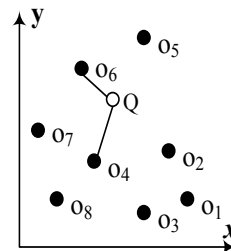
INTRODUCTION

With the rapid development of mobile Internet and the proliferation of geo-positioning like GPS and Wi-Fi, accurate user location is increasing available. Likewise, the numbers

*(Produces the permission block, and copyright information). For use with mlbs2011.cls. Supported by ACM.

of spatial web objects are increasing and available that are being associated with geographic location-specific information and descriptive text documents, such as documents describing restaurants, accommodations, and entertainments at certain regions. This development gives prevalence to spatial keyword queries [1][2][3][4], especially, top- k spatial queries [1][2]. Such a typical query taking a location and a set of keywords as parameters will retrieve a single spatial web object that completely match the keywords. Actually, users usually prefer to submitting spatial keyword queries containing fuzzy terms such as *low*, *moderate* price, *comfortable* environment, and *near* Bird Nest in daily life.

To the best of our knowledge, most existing works on spatial keyword queries that solely regard all labels as equal and seldom focus on how to process the fuzzy labels, and even ignore users preference in a query. However, this type of queries often contain fuzzy labels (e.g., *moderate*, *high*) that can express users preference and intention in a way[5]. In reality, for many scenarios, spatial keyword search for retrieving fuzzy matching is more interesting and coincides with users daily customs. For example, a tourist giving a spatial keyword query (e.g., top- k spatial keyword query) wants to find some hotels near zhongguancun with *moderate* price, while the answering results maybe are irrelevant to his query preference and intention. To help the tourist find the satisfying top- k hotels, we should treat *moderate* as a constraint on the focal keyword *hotel* instead of an independent keyword. In other words, the fuzzy label *moderate* plays the role of a constraint on the spatial keyword queries.



(a) distribution of some restaurants

object	words	$w(a_1)$	$w(a_2)$	$w(a_3)$
o_1	t_1, t_2	0.9	0.8	0.2
o_2	t_1, t_3	0.3	0.5	0.6
o_3	t_2, t_3	0.2	0.6	0.6
o_4	t_3	0.5	0.7	0.8
o_5	t_1, t_4	0.1	0.3	0.3
o_6	t_3, t_4	0.8	0.4	0.4
o_7	t_1, t_2, t_3	0.2	0.5	0.5
o_8	t_2	0.7	0.7	0.2

a_1 :price, a_2 :service quantity, a_3 :popularity

(b) description of objects in Figure 1(a)

Figure 1. Example of PkSK query

In this paper we show how to support spatial keyword queries with fuzzy labels on spatial object datasets. We propose a new kind of query, called preference-based top- k spatial key-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MLBS 11, September 18, 2011, Beijing, China.

Copyright 2011 ACM 978-1-4503-0928-8/11/09...\$10.00.

word (**PTkSK**) queries, which return top-k nearest neighbor objects satisfying users' preference and needs. In the example of Figure 1, a dataset containing the distribution of different restaurants is shown in Figure 1(a). Figure 1(b) depicts the textual and attribute information of restaurants. Without loss of generality, we assume that the weights of object attributes are normalized in the interval [0,1] by using mapping functions. A user submits a query Q that aims to find low price and good service quality restaurant near Zhong-guancun. The objects o_3, o_4, o_6, o_8 marked with gray shade in Figure 1(b) match the keywords of query Q . If we do not consider the user's preference, the existing spatial keyword search techniques can rank o_6 as top-1 result since the object is relevant and closest to Q . However, o_4 is more attractive than o_6 since this object not only satisfies users' preference but also is close to the query location. However, the current works cannot give an efficient support for **PTkSK** queries. In order to process the **PTkSK** queries efficiently, we firstly quantize fuzzy labels by a transforming strategy into the interval [0,1]. And then we propose two efficient searching methods, MFA that is a threshold-based algorithm, and ATRA that is an AIR-tree-based algorithm, to find the objects for meeting users' requirements. The contributions of this work are summarized as follows.

- We propose a new type of preference-based top-k spatial keyword (**PTkSK**) queries, which find the top-k spatial objects ranked by the aggregate score function. The aggregate score function is a linear combination of location proximity, preference similarity and text relevance.
- We propose an efficient transforming strategy that quantizes fuzzy labels into the interval [0,1], and develop a mapping function for projecting each attribute of objects into the [0,1] interval.
- Two query processing algorithms are presented to answer **PTkSK** queries. MFA algorithm employs threshold algorithm to retrieve top-k objects. To reduce computation overhead, we propose ATRA algorithm that is based on an effective hybrid indexing structure called AIR-tree (Attribute Inverted Cell R-tree) for query processing. While AIR-tree extends the R-tree to summarize textual information and attribute vector.
- We conduct extensive experiment to evaluate the scalability and efficiency of our algorithms using synthetic data sets and real-world data sets.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 formally defines problems and concepts. Section 4 introduces the preference computation. In Section 5, we present our proposed hybrid index structure, and Section 6 presents two **PTkSK** queries algorithms. Finally, Section 7 concludes the paper.

RELATED WORK

Spatial keyword queries have received major attention in the recent literature. Various queries on spatial database are studied [1][2][3][4][5][9]. Hariharan et al. [1] was an early

attempt to solve a type of spatial keyword query with range constraints. The objects returned are required to interest with the query MBR and contained all the query keywords. It exploits KR*-tree that extends R*-tree with inverted Cells by augmenting each node of R-tree with the list of all the keywords appearing in the objects of that sub-tree. A similar query is presented in [2], which utilizes a hybrid index namely IR2-tree of R-tree and signature Cells for query processing. m-closest keyword (*mCk*) query [5] aims to return m tuple that cover minimum diameter and match all query keyword. It exploits bR*-tree index which combines R*-tree and bitmap. Besides, each node is augmented keyword MBR to set up more powerful pruning rules. Approximate string match queries [9] in spatial database deal with the issue of simply spelling error or the string containing some errors. Cong et.al [3] propose a location-aware top-k text retrieval (*LkT*) query that employ IR-tree to retrieve the objects both spatial proximity and textual relevance. IR-tree also combines inverted Cells and R-tree. Each non-leaf nodes are augmented with an inverted Cell with an inverted Cell for the objects contained in the sub-tree rooted at the node. Different above hybrid index, this index structure makes the process of search and object ranking simultaneously, which aids to early stop query processing when top-k objects are returned. In our work, we design a hybrid index AIR-tree that extend IR-tree with object attribute vector.

Another area related to our work is feature-based spatial objects queries. Xia et.al [13] address the problem of finding top-k spatial sites based on their influence on feature points, where the influence of spatial site o is defined as the sum of the score of all feature point having o as their closest neighbor. The optimal location queries related to [13] are studied in [14][15]. The aim of [14] is to find the location with the maximal influence whereas [15] retrieves the location that minimizes the average distance between each feature point to its nearest site. Different from [13], the optimal location returned by [14][15] can be any point in the data space while not necessarily an object of the dataset. In addition, Yiu et.al [16][17] propose top-k spatial preference queries that computing the score of a spatial object based on feature objects in its spatial neighbor from multiple feature datasets. Our **PTkSK** query differs from the aforementioned queries. First, we treat fuzzy term contained in query as users' preference instead of common keywords. Second, Fuzzy terms are quantized and mapped into non-spatial attribute of objects so that our query take into account both spatial-text relevance and preference similarity. Therefore, the returned top-k objects are more in line with reflects user's intention.

PRELIMINARIES

In this section, we define the problem of **PTkSK** queries and introduce the basic notation that will be used in the rest of this paper. Furthermore, an efficient transforming strategy is presented to quantizes the fuzzy labels in **PTkSK** query.

Problem Statement

Table 1 shows a list of notations used throughout this paper. Following standard notation, we use capital letters for sets

(e.g. D is a set of spatial objects), and lowercase letters for vectors (e.g. t is a term vector).

Table 1. A list of notations

Notation	Description
D	Spatial object dataset
Q	The PTkSk query
o	An spatial object
p	Spatial location
t	Term vector
f	Attribute feature vector
k	The number of query results
R	The set of returned objects
$w(x_i)$	The weight of attribute a_i
$aScore(\cdot)$	Aggregate ranking function for results
$Mapf(t_j)$	Mapping function for quantizing fuzzy term t_j

Let D be a spatial object dataset. Each object in D is presented as a triple $o=(o.p, o.f, o.t)$, where $o.p$ denotes the spatial location information, $o.f$ is the vector of non-spatial feature attribute, and $o.t$ denotes the associated textual descriptor that is treated as a bag of weighted words using vector space model. $o.f$ is represented by a vector in which each dimension corresponds to the value of non-spatial attribute x_i ($1 < i < n$, $0 < w(x_i) < 1$). We assume that each attribute dimension represents a numerical scoring attribute and therefore, x_i is non-negative numerical value that evaluates feature of spatial objects. With loss of generality, we assume that larger score are preferable.

A preference-based top-k spatial keyword (**PTkSK**) queries consist of three parts $Q(p, f, t)$, where $Q.p$ is a location descriptor; $Q.t$ denotes a set of keywords; $Q.f$ is a fuzzy label like *low price*, *good service*, which represents a constraint on $Q.t$ and provides a better expression of users preference. The **PTkSK** queries will retrieve top-k spatial objects that are ranked by the combination of their location proximity, text relevance and preference similarity. Therefore, it is essential for retrieving top-k spatial objects to deCene an appropriate aggregate ranking function. In this work, we Erstly utilize *Jaccard* metric as a similarity measurement to measure the textual relevance (denoted as $RelT(Q.t, o.t)$) between $Q.t$ and $o.t$, as shown in formula (1).

$$RelT(Q.t, o.t) = \frac{\sum_{i=1}^n x_i \times y_i}{\sum_{i=1}^n x_i^2 + \sum_{i=1}^n y_i^2 - \sum_{i=1}^n x_i \times y_i} \quad (1)$$

where $Q.t = \langle x_1, \dots, x_n \rangle$ denotes a query vector consisting of spatial keywords, $o.t = \langle y_1, \dots, y_n \rangle$ is a document vector of object o .

However, we note that users submitting queries often contain their preferences that are expressed through a set of fuzzy labels such as *low*, *moderate*, *high*, etc. It is infeasible to compute preference similarity by directly using labels. To calculate similarity conveniently, we utilize a transforming strategy to project those fuzzy labels into the interval $[0, 1]$. After transforming those labels, we exploit *Cosine* metric function that is used widely in similarity computation to measure

users preferences (denoted as $Ps(Q.f, o.f)$), as shown in formula (2).

$$Ps(Q.f, o.f) = \cos(Q.f, o.f) = \frac{\sum_{i=1}^n c_i \cdot b_i}{\sqrt{(\sum_{i=1}^n c_i^2 \cdot \sum_{i=1}^n b_i^2)}} \quad (2)$$

where $Q.f = \langle c_1, \dots, c_n \rangle$ registers a vector consisting of fuzzy labels in Q quantized by transforming strategy, $o.f = \langle b_1, \dots, b_n \rangle$ is a vector consisting of non-spatial feature attributes quantized in spatial databases. b_i is the mapping values of attribute x_i in the attribute vector $o.f$ of objects.

While, in query Q , users preference is a type of constraint for keywords, for example, *moderate price hotel*. We utilize $TP(Q, o)$ to measure the users preference similarity and textual relevance, which can be any combination of $RelT(Q.t, o.t)$ and $Ps(Q.f, o.f)$.

$$TP(Q, o) = RelT(Q.t, o.t) \otimes Ps(Q.f, o.f)$$

Operator \otimes is used to represent any arbitrary combination function that respects the fact that $TP(Q, o)$ is monotonically increasing with $RelT(Q.t, o.t)$ and $Ps(Q.f, o.f)$. For simplicity, throughout our discussion we use the linear combination to represent the relationship between $RelT(Q.t, o.t)$ and $Ps(Q.f, o.f)$.

$$TP(Q, o) = \alpha RelT(Q.t, o.t) + (1 - \alpha) Ps(Q.f, o.f) \quad (3)$$

where $\alpha \in (0, 1)$ is used to balance the $RelT(Q.t, o.t)$ and $Ps(Q.f, o.f)$.

Formula(3) mainly focuses on textual relevance of spatial objects and users preference of query, while each object is associated with location information. In order to evaluate the overall score (denoted as $aScore(\cdot)$) of the query result, we propose a linear function that can be employed to combine the $TP(Q, o)$ and location proximity. Again, the combination function can be regarded as an overall aggregate function for ranking the query result. And thus the overall score can be deCened as the following formula.

$$aScore(Q, o) = \beta \frac{RelD(Q, o)}{MaxDis} + (1 - \beta)(1 - TP(Q, o)) \quad (4)$$

where $\beta \in (0, 1)$ is a tuning parameter for adjusting the portion of spatial location proximity $RelD(\cdot)$ and $TP(Q, o)$; Euclidian distance $RelD(Q, o)$ between query Q and object o is normalized to a value between 0 and 1 by a constant $MaxDis$, which denotes the maximum distance of paris of distinct objects in D .

We can see that formula(4) indicates the smaller aggregate score is obtained, the better result for the **PTkSK** query. Formally, we deCene the preference-based top-k spatial keyword (**PTkSK**) queries as follows:

DeEnition 1. (PTkSK queries). Given a query Q and a positive integer k , a **PTkSK** query is to retrieve top- k spatial objects R from D such that $|R|=k$ and $\forall o_1, o_2, o_1 \in R, o_2 \in D-R$ that hold $aSorce(o_1) \leq aSorce(o_2)$.

Preference Computation

From the view of spatial keyword queries, top- k spatial queries return a ranked set of the k best spatial objects. Despite the wide range of location-based applications that rely on top- k spatial queries, the result returned do not always satisfy users' needs. One reason for those queries contain fuzzy linguistic labels (e.g., *low*, *moderate*) that often embody users' preferences, while, Fuzzy linguistic labels used in our natural language to describe some concepts (e.g., *price*) that usually have vague values. In our method each numerical value x_i in A is transformed in a fuzzy set of linguistic labels using mapping function, where A denotes an attribute of object. Note that different attributes can be projected using different mapping functions in terms of the nature of the attribute.

DeEnition 2. (Mapping Function). Let X be a universe set of elements. F is called a fuzzy (sub)set of X , if F is a set of ordered pairs: $F = \{(x_i, Mapf_F(x_i)), x_i \in X, Mapf_F(x_i) \in [0, 1]\}$, where $Mapf_F(\cdot)$ is deEned a mapping function that takes the values of x_i in the interval $[0, 1]$.

The linguistic fuzzy labels given by the users are just approximate ones, we consider that linear trapezoidal mapping functions are good enough to capture the vagueness of those fuzzy labels, since it may be unnecessary to obtain more accurate values. The trapezoidal mapping function can be deEned as the following that is a 4-tuple (l_j, ml_j, mr_j, r_j) .

$$Mapf_F(x_i) = \begin{cases} 1 - \frac{ml_j - x_i}{l_j}, & \text{if } ml_j - l_j \leq x_i \leq ml_j \\ 1, & \text{if } ml_j \leq x_i \leq mr_j \\ 1 - \frac{x_i - mr_j}{r_j}, & \text{if } mr_j < x_i \leq mr_j + r_j \\ 0 & \text{otherwise} \end{cases}$$

where $ml_j \leq mr_j$, $l_j > 0$ and $r_j > 0$ denote the distance of the support of a function to the left and right of ml_j and mr_j , ml_j and mr_j indicate the interval in which the mapping value is 1.

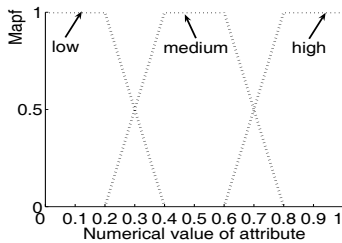


Figure 2. Mapping functions for fuzzy terms: low, medium, high

{*low*, *medium*, *high*} introduced in [19] are three categories of linguistic labels that can cover most of these used in natural language, that is, we can discretize most attributes of objects into the three categories. In this paper, the domain of each attribute of spatial objects is homogeneously divided into three

linguistic labels {*low*, *medium*, *high*} interpreted as mapping functions with trapezoidal shape, as shown in Figure 2. After the fuzzification process, each attribute x of objects is represented by the $\{Mapf_{low}(x_i), Mapf_{medium}(x_i), Mapf_{high}(x_i)\}$, and the vector $\langle o.f(x_1), o.f(x_2), \dots, o.f(x_n) \rangle$ will be represented as $\langle b_1, b_2, \dots, b_n \rangle$ where $b_i = Mapf_F(x_i)$, $b_i \in [0, 1]$.

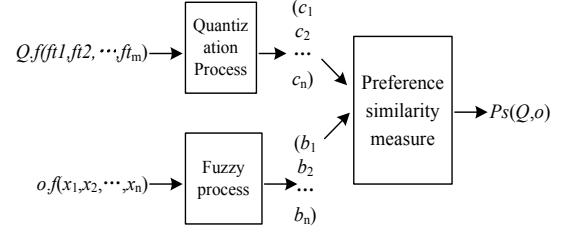


Figure 3. The process of preference computation

Due to fuzzy linguistic labels existing in spatial keyword queries, for example *moderate* price hotel, if we directly utilize crisp values (e.g., 1 or 0) to present the linguistic labels, the results returned seldom satisfy users' preferences in some applications. In this work, fuzzy set theory suggested that **PTkSK** queries contain linguistic labels that are represented as fuzzy variables. Therefore, we have to transform those linguistic labels into numerical value as vectors for calculating the users' preference $Ps(Q, o)$. To process the fuzzy linguistic labels with spatial keyword queries, in this paper we exploit a quantization strategy, called linguistic 2-tuple (t_i, c_i) model that has been presented in [20], to represent the linguistic labels information, where t_i is a linguistic label (e.g. *moderate*) and c_i is a numeric value in the $[0, 1]$ interval.

DeEnition 3. (Quantization Process). Let $S = \{l, \dots, l_k\}$ and $S_T = \{s_1, \dots, s_g\}$ be two linguistic labels sets, such that $g \geq k$, where l_i denotes the linguistic labels given by users, s_i denotes the predefined linguistic labels. Quantization process including two steps $T(S \rightarrow S_T)$ and $2T(S_T \rightarrow (l_i, c_i))$ are deEned as:

$$\begin{aligned} T(\cdot) : S &\rightarrow S_T \\ T(l_i) &= \{(s_k, r_k), k \in \{0, \dots, g\}\} \\ r_k &= \max \min \{Mapf_{l_i}, Mapf_{r_k}\} \\ 2T(\cdot) : S_T &\xrightarrow{2\text{-tuple model}} (l_i, c_i) \end{aligned}$$

Figure 3 shows the process of preference computation. Once the two vectors $\langle c_1, \dots, c_n \rangle$ and $\langle b_1, \dots, b_n \rangle$, are given, we can adopt the formula (2) to calculate the users' preference $Ps(Q, o, f)$.

AN EFFICIENT HYBRID INDEXING

In this section, we present an efficient hybrid index structure, called AIR-tree (Attribute Inverted Element R-tree [7]), which is based on the IR-tree [2]. AIR-tree clusters spatially close objects together, and carries textual information and attribute vectors in one node. The attribute vectors are used in users

preference similarity computing. The AIR-tree therefore can improve searching efficiency of **PTkSK** queries.

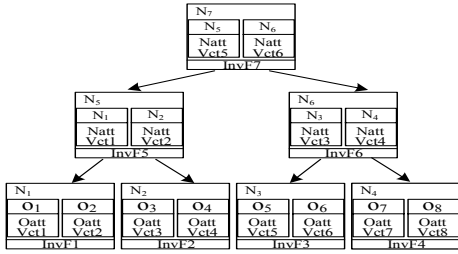


Figure 4. The AIR-tree of Figure 1

Table 2. Content of Inverted Files of Leaf nodes in AIR-tree

Node	N_1	N_2	N_3	N_4
Inverted File	$t_1 : o_1, o_2$ $t_2 : o_1$ $t_3 : o_2$	$t_2 : o_3$ $t_3 : o_3, o_4$	$t_1 : o_5$ $t_3 : o_5$ $t_4 : o_5, o_6$	$t_1 : o_7$ $t_2 : o_7, o_8$ $t_3 : o_8$
Att vector (a_1, a_2, a_3)	0.3, 0.8, 0.6	0.2, 0.7, 0.8	0.1, 0.4, 0.4	0.2, 0.7, 0.5

In the AIR-tree, a leaf node includes some entries of the form ($Optr, Loc, Odi, OattVct$), where $Optr$ denotes a pointer to an object in the database, Odi expresses the identifier of the document of object. $OattVct$ registers the attribute vector of object. The weight of each item (denoted as $w(a_i)$) in $OattVct$ is the value of corresponding attribute a_i ($0 < w(a_i) < 1$). A non-leaf node contains these entries of the form ($Nptr, MBR, Ndi, NattVct$), $Nptr$ denotes the pointer to the child nodes, Ndi represents an identifier of a pseudo text description that is the summary of all text description in the entries of child node. $NattVct$ is a pointer to the node attribute vector where the weight of each attribute is the maximum weight value contained in the subtree rooted at $Nptr$. Figure 4 illustrates an AIR-tree example for the dataset in Figure 1. The inverted file and attribute vector of non-leaf node N_1, N_2, N_3, N_4 are only shown in Table 2. For example, each item in $InvF_1$ is the union of the terms in the document contained in o_1 and o_2 . The weight of attribute of node N_1 is (0.3, 0.8, 0.6), respectively, which are minimum weight of a_1 and maximum weight of a_2 and a_3 in the two object attribute vectors $OattVct_1$ and $OattVct_2$. The maintenance of AIR-tree exploits the insert operation, which is adjusted the original R-tree operation by appending the operations of updating inverted file and attribute vector when **AdjustTree()** is invoked. Similarly, the operations of update and delete in AIR-tree can also be extended according to the corresponding R-tree operations.

PROCESSING OF PTKSK QUERIES

In this section, we introduce two query processing algorithms. One of which is a threshold-based method to find **PTkSK** result set; the other one is based on the hybrid index structure to process **PTkSK** queries.

MFA Algorithm

In this subsection, we propose a **PTkSK** query processing method, called *Multiple feature algorithm* (denoted as MFA),

which is based on the *threshold algorithm* (denoted as TA). TA algorithm [6] is a typical method to addressing top-k query that returns k tuples with the highest scores according to a monotone function. We would partition the **PTkSK** query into three features, that is, query location represents spatial feature sf , fuzzy constrains and query keywords respectively corresponds to users preference feature pf and text feature tf . Therefore, the submitted query Q is transformed into a set of three features $Q_f(sf, pf, tf)$. To keep the same monotony of the three features, we can do adjustment in some extent. We assume that the aggregate function is monotone decreasing in this algorithm. Similarly, we adjust the TA algorithm to answer **PTkSK** query. The pseudo-code of MFA is shown in algorithm 1.

Algorithm 1. MFA (Q, k);

Input: Q : a **PTkSK** query;

k : a positive number of returned results;

Variables: GT : a global threshold;

$BaScore$: best aScore that aggregates the score of the current best objects.

Output: R : The top-k objects satisfying Q ;

```

1:  $Q_f \leftarrow \text{Transform}(Q)$ ;
2:  $GT \leftarrow 0$ ;
3:  $BaScore \leftarrow \infty$ ;
4:  $R \leftarrow \text{Null}$ ;
5: for each each feature  $q_i$  in  $Q_f$  do
6:    $t_i \leftarrow 0$ ;
7: for  $i$  from 1 to  $k$  do
8:   while( $GT < BaScore$ ) do
9:     for  $i$  from 1 to 3 do
10:      select query feature  $q_i$ ;
11:      get the match  $o_j$  of  $q_i$ ;
12:       $t_i \leftarrow \text{score of } o_j \text{ on feature } q_i$ ;
13:      update  $GT$ ;
14:      if  $GT \geq BaScore$  then
15:        break;
16:      compute  $aScore(o_j, Q_f)$ 
17:      if  $aScore(o_j, Q_f) < BaScore$  then
18:        cur-bestresult  $\leftarrow o_j$ ;
19:         $BaScore \leftarrow aScore(o_j, Q_f)$ ;
20:    $R \leftarrow R \cup \{o_j\}$ ;
21: return  $R$ 

```

The algorithm begins with transforming Q into Q_f in the form of (sf, pf, tf) and quantizes pf in transforming strategy (line 1). Line 7-21 illustrates the iterative process to find the top-k spatial objects. In query processing (line 8-19), we begin to retrieve the first nearest neighbor objects o_1 of the spatial feature q_1 (line 10-11) and compute aggregate score between o_{i1} and Q (line 16), and then we find the objects o_2 with highest preference similarity score and o_{i3} with highest text relevance score (line 10-11), based on which, we compute the corresponding aggregate score in round-robin fashion (line 16). After finishing one loop, we obtain current total thresholds GT (line 12-13) and current best results (line 17-19). If $GT < BaScore$, then there exists some object in D whose aggregate score is smaller than $BaScore$. Otherwise,

we can retrieve the top-1 object. The algorithm terminates when top-k spatial objects are found.

ATR-tree-based Algorithm

MFA algorithm may incur multiple accesses to the same nodes and retrieve the same data point through different queries. To overcome this drawback, we depend on the AIR-tree to retrieve those objects only containing some query keywords and satisfying users' preference, which can avoid checking the irrelevant objects to query. In order to process **PTkSK** queries efficiently, AIR-tree is traversed from the root node following the best-First traversal strategy. Algorithm2 shows the pseudo-code of **PTkSK** queries.

Algorithm 2. ATRA (Q, T, k);

Input: Q : a **PTkSK** query;
 T : an AIR-tree;
 k : a positive number of returned results.
Output: R : the top-k objects satisfying Q ;

- 1: $Q_f \leftarrow \text{Quant}(Q)$;
- 2: $U \leftarrow$ new min-priority queue;
- 3: $U.Enqueue(T.root, 0)$;
- 4: **while** U is **not** empty **do**
- 5: $E \leftarrow U.Dequeue()$;
- 6: **if** E is an object **then**
- 7: $R \leftarrow R \cup E$;
- 8: **if** $|R| = k$ **then**
- 9: **goto** 16;
- 10: **else if** E is an intermediate node **then**
- 11: **for** each entry e in E **do**
- 12: $U.Enqueue(e, MINaScore(e, Q_f))$;
- 13: **else if** E is a leaf node **then**
- 14: **for** each object o in E **do**
- 15: $U.Enqueue(o, aScore(o, Q_f))$;
- 16: **return** R ;

This algorithm maintains a priority queue U to store the AIR-tree nodes and objects that have yet to be visited, where we sort nodes or objects in descending order of their relevance with respect to query Q . As an importance metric, the $MINaScore$, is defined to provide the lower bound of the relevant aggregate score between query Q and the objects enclosed in the rectangle of node N , which can be used to order intermediate node and efficiently prune the paths of the search space in hybrid index. The metric $MINaScore$ can be estimated as follows:

$$MINaScore(Q, N) = \beta \frac{\min RelD(Q.loc, N.mbr)}{MaxDis} + (1 - \beta)(1 - TP(Q, N))$$

where β is the same as in formula (4), $\min RelD(Q.loc, N.mbr)$ denotes minimum Euclidian distance between $Q.loc$ and $N.mbr$; $TP(Q, N) = \alpha RelT(Q.t, N.t) + (1 - \alpha) Ps(Q.f, N.f)$.

We can see that $MINaScore(Q, N)$ provides a lower bound on the actual aggregate score between **PTkSK** query and the objects enclosed in the rectangle of node N . Therefore, we proceed to present the following pruning rule.

Lemma1: Let $CMINaScore$ is the current smallest aggregate score of node. The node N can be safely pruned if $CMINaScore \leq MINaScore(Q, N)$.

Proof: Since $MINaScore(Q, N)$ is the lower bound of all nodes in the subtree rooted at node N , the aggregate score of any object contained in $subtree(N)$ must be larger or equal than $MINaScore(Q, N)$. Therefore, we can safely prune the node N when $CMINaScore \leq MINaScore(Q, N)$, which avoids traversing these nodes in $subtree(N)$ during query processing.

In this algorithm, we employ **Quant()** function to quantize the fuzzy labels in Q by using transforming strategy for getting Q_f . The query process iteratively checks the First entry E in U (line 4-16). If E is a intermediate node (line 10-12), then all its child nodes enqueue to U together with $MINaScore$ score for later processing. Likewise, if E is leaf node (line 13-15), then all objects in E enqueue to U . Note that algorithm only compute these nodes or objects that need to be traversed in ATR-tree. Once R contains k objects or no more objects can be found, the algorithm terminates and outputs R .

EXPERIMENTS

In this section, we experimentally evaluate the efficiency and scalability of our proposed algorithm on synthetic and real datasets.

Experimental Setup

The synthetic dataset (SN) is generated to resemble the real world. The synthetic data generator generates spatial data points in a random manner. All data points are distributed 2-dimensional space and obey uniform distribution. Each object is assigned four attribute a_1, a_2, a_3, a_4 , and combines a document that randomly selects from WEBSpAM-UK 2007 that consists of a large of real web document. The attribute score of each object is uniformly generated within the range $[0, 1]$. The real dataset (RL) contains information obtained by the online web data resource PocketGPSWorld[18]. The object dataset consists of point of interests (POI) which belong to restaurant in the region of United States. Each object contains three non-spatial attributes, price, service quantity and popularity respectively. These attribute values are generated randomly range from 0 to 1. The properties of two datasets are shown in Table2.

Table 3. Datasets details

Dataset	Total objects	Total unique words	Total words
SN	2,631,576	1,184,856	602,630,904
RL	103,258	20,389	17,164,896

Each query contains 2 to 8 keywords, 1 to 4 fuzzy terms and a location from dataset SN and RL. The hybrid structure AIR-tree is disk-resident. The page size is fixed at 4KB and the number of entries in each node is 136. All the experiments are conducted in Java and run on a Windows platform with an Intel(R) core(TM) 2 Quad CPU Q8400 @2.66GHZ with 4GB RAM. In this paper, we adopt the running time as the main metric to measure the performance of our proposed

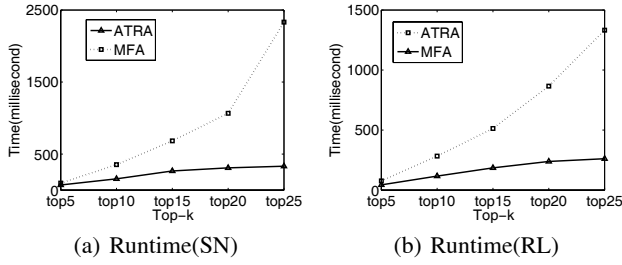


Figure 5. Effect of the k value

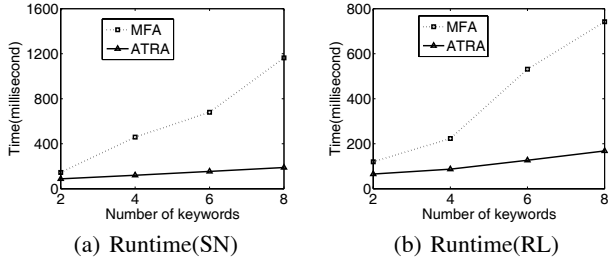


Figure 6. Effect of the number of keywords

algorithms. The experiment setting is as follows:
Number of results k : 5 to 25, default 10
Number of query keyword: 2 to 8, default 2
Number of query fuzzy term: 1 to 4, default 2
Parameter $\alpha, \beta: 0 \sim 1$, default 0.4

Experiment results

We compare the performance of the ATR-tree based algorithm (ATRA) and multiple feature algorithms (MFA). Two sets of experiments are carried out. The first set of experiments measure the efficiency of the algorithms for varying the different parameters introduced in section 5.1. The second set of experiments measure the scalability of the algorithms for varying the dataset size.

Efficiency In the first experiment, we use our default setting and study the running time for all datasets while varying k . The results in Figure 5 show that ATRA algorithm performs better than MFA algorithm for all values of k , since MFA need to access more objects and potentially more intermediate nodes in order to report the top- k results. On the contrary, ATRA can prune these nodes irrelevant to the query so that the running time is reduced remarkably. As expected, the running time of two algorithms increases as k grows.

Next, we evaluate the performance of the two algorithms with respect to the number of submitted keywords. The results are shown in Figures 6(a) and (b). All algorithms need more time as the number of keywords increases since they need to process more keywords. The ATRA algorithm significantly outperforms MFA algorithm of the different number of keywords. This is because ATRA can prune disqualified node in query processing. On the other hand, the running time of ARTA increases slowly as the number of keywords grows.

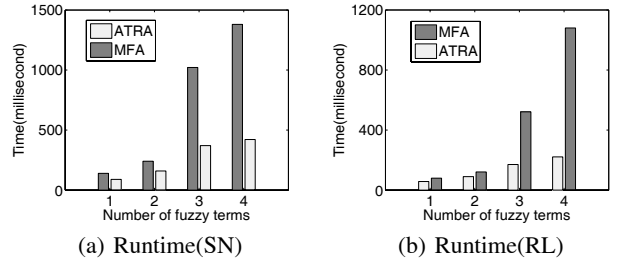


Figure 7. Effect of the number of fuzzy terms

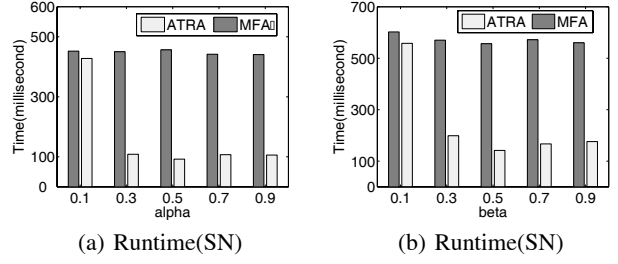


Figure 8. Effect of parameters α and β

Third, we vary the number of fuzzy terms, that is the number of mapping attribute, and evaluate the performance of our proposed algorithm. The results for dataset SN and RL are shown in Figures 7(a) and 7(b). In general, the running time of our algorithm increase with the number of fuzzy term increasing. For all datasets, the performance of ATRA significantly outperforms MFA algorithm. The main reason is that the ATRA exploits attribute summary information in each level of ATR-tree to improve the pruning capabilities, therefore they just need to search a small area of the space for retrieving the best objects. On the other hand, we can observe that two algorithms have similar trends when the number of fuzzy term is small.

In the subsequent experiment, we evaluate the impact of different parameters α and β on synthetic dataset. Figures 8(a) and 8(b) plot the runtime of MFA and ATRA with the two parameters varying from 0.1 to 0.9. Parameter α in formula(3) allows users to adjust the importance between textual relevance and users preference. We fix k at 10, the number of query keywords at 2, β at 0.4, and the number of fuzzy terms at 2. Figure 8(a) shows that runtime of ATRA decreases as α increases. That is because it takes shorter for computing the users preference with a larger α . The performance is the best at $\alpha=0.5$. While α seldom impacts the runtime of MFA, the reason is that in MFA spatial distance textual relevance and users preference are not combination but are considered respectively. Figure 8(b) shows the changes of runtime of MFA and ATRA. Parameter β in formula(4) allows users to balance the importance among spatial distance, textual relevance and users preference. k , the number of query keywords and the number of fuzzy terms are fixed as the same as Figure 8(a), while α is fixed at 0.4. In Figure 8(b), a large β denotes that spatial distance is more important, while a small β denotes that textual relevance and

users preference are more important. The performance is the best at $\beta=0.5$.

Scalability In the second set of experiment, we study the scalability of our proposed algorithm against dataset size. Different synthetic datasets that contains the number of objects from 2million to 10 million, are generated by data generator. The corresponding document of each object is assigned a document randomly from the WEBSpAM-UK 2007. The results are shown in Figure8. We can see that the scalability of ATRA is superior to MFA obviously. On the contrary, the MFA does not scale with increasing the number of objects since it is sensitive to the dataset size. Also, note that ATRA results in small running time even in the case of the number of objects is 10 million.

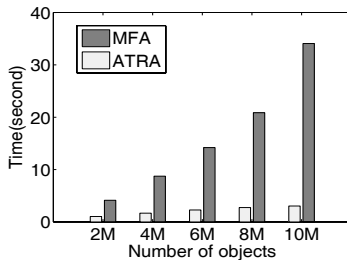


Figure 9. Effect of the dataset size

CONCLUSIONS

The paper presents a comprehensive study for preference-based top-k spatial keyword (**PTkSK**) queries. We exploit an efficient strategy of computing users preference to transform fuzzy linguistic labels into the $[0,1]$ interval. We present an efficient hybrid indexing structure AIR-tree that extends the R-tree using inverted index and feature attribute vector. We propose two query processing algorithms to answer the **PTkSK** query. We conducted extensive performance evaluations for the proposed algorithms by using both real and synthetic data sets.

ACKNOWLEDGMENTS

This research was partially supported by the grants from the Natural Science Foundation of China (No.60833005, 61070055, 91024032), the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (No. 10XNI018), National Science and Technology Major Project of Key Electronic Devices, High-end General-purpose Chips and Fundamental Software Products (No. 2010ZX01042-002-003), Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 200800020002)

REFERENCES

1. R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial keyword (sk) queries in geographic information retrieval systems. In *Proc. of the 19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, 2007.
2. I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *Proc. of the 24th International*

3. *Conference on Data Engineering (ICDE 2008)*, pages 656 - 665, 2008.
3. G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Journal of Proc. of VLDB Endowment (PVLDB 2009)*, 2(1): 337-348, 2009.
4. Y. -Y Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. of the 26th International Conference on Management of Data (SIGMOD 2006)*, pages277-288, 2006.
5. D. X. Zhang, Y. M. Chee , M.Mondal, A. K. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *Proc. of the 25th International Conference on Data Engineering (ICDE 2009)*, pages 688-699, 2009.
6. D.X. Zhang, B. C. Ooi, and A.K. H Tung. Locating mapped resources in web 2.0. In *Proc. of 26th International Conference on Data Engineering (ICDE 2010)*, pages521-532, 2010.
7. S.K.M. Wong, W. Ziarko, V.V. Raghavan, P.C.N.Wong . On modeling of information retrieval concepts in vector space. *ACM Transaction on Database System(TODS 1987)*,12(2):299-321,1987.
8. V.N.Anh,O.de Kretster, and A.Moffat. Vector space ranking with effective early termination. In *Proc. of the 24th International Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 35-42, 2001.
9. B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou. Approximate string search in Spatial Databases. In *Proc. of the 26th International Conference on Data Engineering (ICDE 2010)*, pages 545-566, 2010.
10. R. Fagin,A. Lotem, and M. Naor. Optimal aggregation algorithm for middleware. *Journal of Computer System and Sciences*, 66(4):414-656, 2003.
11. A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *Proc. of the 4th International Conference on Management of Data (SIGMOD 1984)*, pages 47-57, 1984.
12. J.Zobel, and A.Moffat. Inverted CEs for text search engines. *ACM Computing Surveys*. 38(2):6, 2006.
13. Y. Du, D. Zhang, and T. Xia. The optimal-location query. In *Proc. of the International Symposium on Spatial and Temporal Databases (SSTD 2005)*, pages 163-180, 2005.
14. T. Xia, D. Zhang,E. Kanoulas, and Y. Du. On computing top-t most incoventional spatial sites. In *proc. of 31th International Conference on Very Large Data Bases(VLDB2005)*, pages 946-957,2005.
15. D.Zhang, Y.Du,T.Xia, and Y. Tao. Progressive computation of the min-dist iptimal-location query. In *proc. of 32th International Conference on Very Large Data Bases(VLDB2006)*, pages 643-654,2006.

16. M.L. Yiu, X.Dai, N. Mamoulis, and M. Vaitis. Top-k spatial preference queries. In *Proc. of 23th International Conference on Data Engineering (ICDE 2007)*, pages 1076-1085, 2007.
17. M.L. Yiu, X.Dai, N. Mamoulis, and M. Vaitis. Ranking spatial data by quality preference. *Transactions on Knowledge of Data Engineering (TKDE2011)*, 23(3):433-446.
18. PocketGPSWorld, www.pocketgpsworld.com/modules.php.
19. F. Herrera and L. Martinez. A 2-tuple fuzzy linguistic representation model for computing with words. *Transactions on Fuzzy Systems (TFS2000)*, 8(6):746-752.
20. F. Herrera, L. Martinez and P. J. Sanchez. Managing non-homogeneous information in group decision making. *European Journal of Operational Research*, 166(1):115-132.