International Journal of Data Warehousing and Mining
Volume 19 • Issue 2

An Efficient Association Rule Mining-Based Spatial Keyword Index

Lianyin Jia, Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China
https://orcid.org/0000-0002-0269-9017

Haotian Tang, Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China

Mengjuan Li, Library, Yunnan Normal University, China

Bingxin Zhao, Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China

Shoulin Wei, Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China*
https://orcid.org/0000-0002-3547-4025

Haihe Zhou, Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China

ABSTRACT

Spatial keyword query has attracted the attention of many researchers. Most of the existing spatial keyword indexes do not consider the differences in keyword distribution, so their efficiencies are not high when data are skewed. To this end, this paper proposes a novel association rule mining based spatial keyword index, ARM-SQ, whose inverted lists are materialized by the frequent item sets mined by association rules; thus, intersections of long lists can be avoided. To prevent excessive space costs caused by materialization, a depth-based materialization strategy is introduced, which maintains a good balance between query and space costs. To select the right frequent item sets for answering a query, the authors further implement a benefit-based greedy frequent item set selection algorithm, BGF-Selection. The experimental results show that this algorithm significantly outperforms the existing algorithms, and its efficiency can be an order of magnitude higher than SFC-Quad.

KEYWORDS

ARM-SQ, Greedy Frequent Itemset Selection, Inverted Index, Materialized View, Quadtree, SFC-Quad, Spatial Keyword Query, Z-Curve

INTRODUCTION

The rapid development of mobile devices has produced many location-based services, and an increasing number of online objects have both spatial and textual properties. For example, social services such as Twitter and Foursquare allow users to send their tweets with location; navigation services such as Google Maps and Baidu Map allow users to search for points of interest nearby. These applications require efficient spatio-textual indexes to support numerous spatial keyword queries.

DOI: 10.4018/IJDWM.316161 *Corresponding Author

There are three common spatial keyword queries: Boolean range query (BRQ), Boolean *k*NN query (B*k*Q) and the top-*k* *k*NN query (T*k*Q) (Chen et al., 2013). This paper focuses on BRQ, which is fundamental to many spatial textual applications (Salgado et al., 2018; Tampakis et al., 2021; Wang et al., 2021; Wang et al., 2020). Given a collection of spatio-textual data, a spatial keyword query q (comprising a spatial region R and a set of keywords W), BRQ aims to find all objects, each of which is in R and contains all keywords in W .

In Figure 1, a user may want to find a restaurant in a certain area (query region in a dashed rectangle) that sells hamburgers and coffee (query keywords) together. He can issue his query to a BRQ server and obtain point 1 as the returned result. Although points 3 and 4 are also in this area, they do not contain all the query keywords. For example, P2 sells hamburgers and coffee together, but they are too far away.

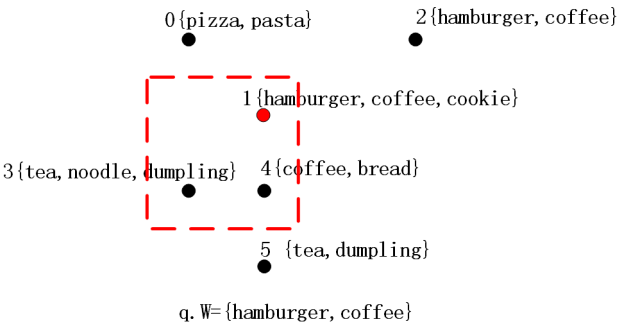
The spatial keyword index is the heart of an efficient spatial keyword query. We can consider it a spatial and textual index combination, either in a space-first or text-first way. Based on the spatial indexes used, existing studies can be R-tree-based, grid-based, and space-filling curve-based. From the textual perspective, we can also classify these studies into inverted index-based and bitmap-based studies. Chen et al. (2013) experimentally evaluated 12 spatial-textual indexes, such as IR-Tree (Wu et al., 2012b), WIBR-Tree (Wu et al., 2012a), and SFC-Quad (Christoforaki et al., 2011), and the results show that SFC-Quad performs the best both in query time and space on BRQ.

The excellent performance of SFC-Quad is because of the simple core structure: an inverted index ordered by space-filling curves. The objects in each inverted list are sorted according to the Z-curve order, so the objects close in the original space are as close as possible in the inverted lists, thus leading to a better query performance.

Although many studies followed the study by Chen et al. (2013) into BRQ, most turn to in-memory or streaming data spatial keyword query (Chen et al., 2017; Mahmood et al., 2018), collaborative spatial keyword query (Zhao et al., 2017), semantic spatial keyword query (Qian et al., 2018; Sun et al., 2017), privacy-preserving spatial keyword query (Cui et al., 2019; Su et al., 2015), and spatial keyword query on road network (Han et al., 2015; Zheng et al., 2016). We can see from recent surveys (Chen et al., 2020; Chen et al., 2021) that SFC-Quad is still the best representative in disk-based BRQ algorithms.

However, most of the existing indexes do not consider the differences in keyword distribution, so the efficiency is not high when data are skewed. Skewed distribution is ubiquitous in real scenarios. A large variety of keywords appear only in a few objects, while a few appear in many objects. Therefore, treating all keywords equally will inevitably lead to a decrease in query efficiency. The experimental results of this paper show that the query time of SFC-Quad increases sharply with the increase in frequent words. Therefore, it is necessary to consider effective measures to improve query efficiency under a skewed distribution. Although WIBR-tree has considered the skewed distribution,

Figure 1. A BRQ Example



its iterative bipartition manner on the most frequent word cannot fully make use of the characteristics of the skewed distribution.

Considering that list intersections are the most expensive operations in SFC-Quad and materialization is an effective way to improve the efficiency, this paper designs a novel association rule mining (ARM) based spatial keyword index, ARM-SQ. The inverted lists in ARM-SQ are materialized by the mined frequent itemsets (FIS); thus, intersections of long lists can be avoided. A depth-based materialization strategy is introduced to prevent excessive space costs caused by materialization, which maintains a good balance between time and space costs. To select the right FISs for answering a query, we further implement a benefit-based greedy frequent itemsets selection algorithm, BGF-Selection. The experimental results show that our algorithm significantly outperforms existing algorithms, and its efficiency can be an order of magnitude higher than SFC-Quad.

The contributions of this paper are as follows:

1. To the best of our knowledge, this is the first study combining ARM and spatial keyword query to improve the performance of high-frequency words.
2. A novel depth-based materialization strategy is proposed, which maintains a good balance between time and space costs. As a result, ARM-SQ can decrease query time significantly by introducing only small space overheads.
3. BGF-Selection, an efficient benefit-based greedy frequent itemset selection strategy, is designed, which can quickly select an approximate optimal frequent itemset combination to answer the query.
4. Extensive experiments on multiple real datasets show that the proposed algorithm significantly outperforms existing algorithms. Furthermore, our algorithm runs up to an order of magnitude faster than SFC-Quad and has a better scalability with increasing the query range.

The rest of this paper is organized as follows. We next introduce the problem definitions and necessary preliminaries. Next, our core ARM-SQ index and the corresponding algorithm are presented before we present the experimental results. We then conclude the paper.

PROBLEM DEFINITION AND PRECONDITION

Basic Definition

Given a spatial keyword dataset *D*, each object $o \in D$ is defined as $o = \langle o.C, o.W \rangle$, where $o.C$ is a coordinate consisting of longitude and latitude and $o.W$ is a set of keywords. Each object o has a unique identifier $o.ID$. In this paper, $|D|$ is denoted as the number of objects in the dataset, $|o.W|$ is denoted as the number of elements in $o.W$ and $o.W[i]$ is denoted as the i-th element in $o.W$ ($0 \le i \le |o.W| - 1$). A spatial keyword query has a form of $q = \langle q.R, q.W \rangle$, where $q.R$ is a rectangular query region bounded by the upper-left and lower-right coordinates. $q.W$ is the corresponding set of query keywords.

Based on these discussions, the definition of spatial keyword query is given as follows:

Definition 1: Spatial keyword query: Given a query $q = \langle q.R, q.W \rangle$, a BRQ retrieves all objects, each of which is in $q.R$ and contains all keywords in $q.W$.

Spatial Keyword Query

As a building block for many spatial keyword-based applications, BRQ has been extensively researched for a long time. Among all the indexes, R-tree based indexes are the most popular. IR-tree is a typical

representative that combines an R-tree and an inverted index. Based on IR-tree, WIBR-tree and many other variants have been proposed.

In WIBR-tree, the objects are first partitioned into two groups using the most frequent word *w1*. All objects in the first group contain *w1*, and no object in the second group contains *w1*. Each of these two groups is then further iteratively partitioned by the next frequent word until the number of objects in each group is below a certain threshold. We use the partitioned groups as leaf nodes to build the IR-tree in a bottom-up way. Although WIBR-tree has considered the skewed distribution of keywords, which is like the main idea of this paper, its iterative bipartition manner on the most frequent word cannot fully make use of the characteristics of a skewed distribution. In contrast, our algorithm exploits more frequent information in the dataset by employing ARM.

A space-filling curve (SFC) can also accelerate BRQ, and a SFC can transform high-dimensional data into 1-dimensional or sort high-dimensional data (Jia et al., 2021). SFC-Quad is an application of the latter case, and it sorts objects by Z curve in ascending order and inserts them in inverted lists used in sorted order, thus having high efficiency.

A memory-based BRQ was studied by Lee et al. (2015), which employs five memory-based optimizations to accelerate in-memory BRQ. Tampakis et al. (2021) mapped space to distance and text to similarity, reducing spatial text data to 2-dimensional, and then studied BRQ on the mapped data. However, their work is an approximate query from the perspective of the text. Wang et al. (2020) studied BRQ in encrypted data, and the index they used is a combination of SFC (gray code) and quadtree. For more research in this area, readers are referred to recent research surveys (Chen et al., 2020; Chen et al., 2021; Gao & Jensen, 2016).

Different from the studies above, this paper aims to design a disk-based algorithm and adopts a novel idea of combining ARM and BRQ, which can greatly improve the query performance of existing algorithms, especially when the query contains a large number of frequent words.

Association Rule Mining

ARM is a classic problem studied in the data mining community for a long time. The core of ARM is to efficiently obtain FISs whose support is greater than a certain threshold. ARM is based on an important property: if an itemset is infrequent, all its supersets are also infrequent.

Apriori (Agrawal & Srikant, 1994), FP-Growth (Han et al., 2000), and Eclat (Zaki, 2000) are the three classic ARM algorithms. Based on the ideas of these algorithms, existing algorithms can be divided into three categories: generation-test-based algorithms, pattern growth-based algorithms, and vertical grid-based algorithms. There is a range of recent improvements to the three classic algorithms (Alghyaline et al., 2016; Zhang et al., 2019; Zhang et al., 2019; Zhu & Liu, 2019; Ding et al., 2022).

This paper generates FISs based on FP-Growth, but other ARM algorithms can also be applied here. For ease of description and understanding, the symbols involved in this paper are listed and defined in Table 1.

THE PROPOSED INDEX

Motivation

Although having high efficiency, SFC-Quad also has the following shortcomings:

1. Keywords are generally skewed. Very few keywords appear in many objects, which corresponds to long lists. Querying long lists requires more IO operations, which are inefficient.
2. For multiple high-frequency words, expensive list intersections are needed, which leads to the degradation of SFC-Quad performance.

Table 1. Symbolic Table

Symbol	Description
D	A spatial keyword dataset
$q = \langle q.R, q.W \rangle$	a spatial keyword query, where $q.R$ is the query region and $q.W$ is the query keywords
$o = \langle o.C, o.W \rangle$	a spatial keyword object, where $o.C$ is the coordinate and $o.W$ is the keywords of o
h	a materialized depth
H	a set of materialized depths
C_t^h	the normalized query time cost for depth *h*
C_s^h	the normalized space cost for depth *h*
t^h, s^h	the query time and space for depth h, respectively
t^{max}, s^{max}	the maximum query time and space cost for any depth in H, respectively
F	the FISs mined for the optimal h
$q.W_f, q.W_n$	the frequent subset and infrequent subset of $q.W$, respectively
f	a FIS
$F^{q.W_f}$	all FISs covered by $q.W_f$
B_f	the benefit of f
S_e, S_f	the support of element e and FIS f, respectively
F^S	the selected FISs after executing BGF-Selection

Considering these shortcomings, in this paper, we design an ARM-based index, ARM-SQ, and the corresponding query algorithm to improve BRQ efficiency.

ARM-SQ

The main parts of ARM-SQ include a materialized inverted index and a quadtree. For a sample dataset in Table 2 (the objects have been sorted in Z ascending order, and we have reassigned the IDs as shown in Figure 2), the corresponding ARM-SQ is shown in Figure 3. Note that although other space-filling curves, e.g., Hilbert curves, can also similarly sort the objects, we choose Z order here as it has fast encoding and decoding speed (Jia et al., 2021).

Table 2. Sample Dataset

o.ID	o.C	o.W
0	(0,0)	{a,b,c,e}
1	(1,1)	{a,b,c,e,f}
2	(3,0)	{b,c,g}
3	(0,2)	{d,g,h}
4	(1,2)	{a,b,d}
5	(1,3)	{a,d,f}

Figure 2. Sorted Objects in Table 1 Using a Z Curve

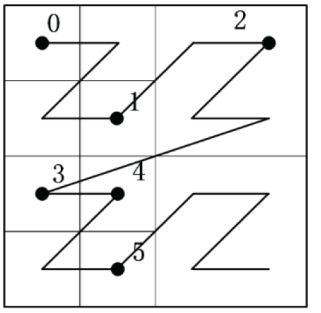
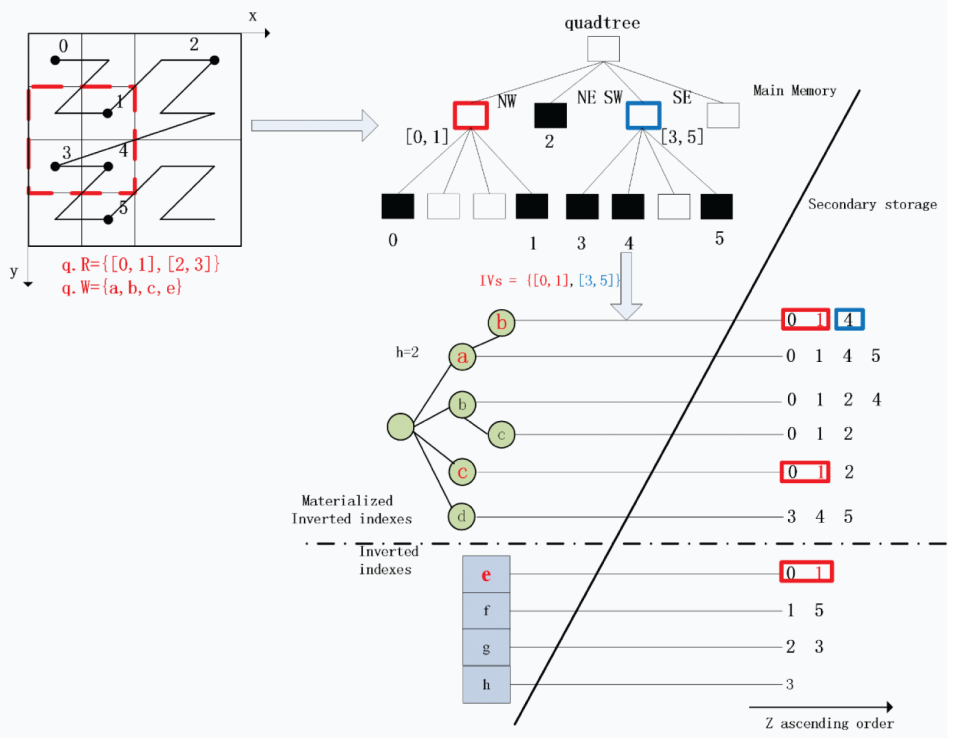


Figure 3. The ARM-SQ Index



Materialized Inverted Index

The heart of ARM-SQ is the materialized inverted index. To effectively build this index, we introduce ARM to generate FISs and materialize the intersections of inverted lists using generated FISs. However, when the support is low, numerous FISs will be generated and materializing them is space prohibited.

This paper designs a depth-based materialization strategy to materialize only a small fraction of inverted lists. To do this, a depth parameter h is introduced to materialize only the FISs whose depth is not more than h, which can effectively control the growth of storage space. Note that the effect of parameter h on materialization is twofold. On the one hand, as h increases, more FISs are materialized, so more list intersection operations can be avoided. On the other hand, as h increases, more storage space is needed. For a better balance between query time and storage space, given a depth set $H = \{h_1, h_2, ..., h_n\}$ containing n materialized depths, we can calculate the space-time comprehensive cost C^h for a depth $h \in H$ according to Equation (1):

$$C^h = C_t^h + C_s^h \tag{1}$$

where C_t^h and C_s^h represent the normalized query time cost and space cost for h, respectively. C_t^h and C_s^h can further be calculated according to Equations (2) and (3) as follows:

$$C_t^h = t^h \,/\, t^{max} \tag{2}$$

$$C_s^h = s^h \,/\, s^{max} \tag{3}$$

In these equations, t^h and t^{max} are denoted as the corresponding query time for depth h and the maximum query time for any depth in H, respectively. Correspondingly, s^h and s^{max} are denoted as the corresponding space costs for depth h and the maximum space cost for any depth in H, respectively.

In this way, the optimal h corresponding to the minimum comprehensive cost can be calculated by Equation (1); then, the corresponding FISs can be determined and used to materialize the inverted lists. For ease of description, we use F to denote the FISs mined for the optimal h.

To effectively organize the mined FISs, we organize them into a trie. Each FIS in F is represented as a path starting from the root in trie. Each node in trie points to a materialized inverted list. The trie resides in memory, while the inverted lists are compressed by OPT-PFD (Yan et al., 2009) and stored in external memory. The elements in inverted lists are sorted in Z ascending order as described before.

Example 1: For the dataset in Table 1, we mine FISs using a support threshold 3 and obtain $F = \{\{a\}, \{b\}, \{c\}, \{a,b\}, \{b,c\}, \{d\}\}$; then, we materialize the intersections for FISs $\{a,b\}$ and $\{b,c\}$, as shown in Figure 3.

Quadtree

A quadtree is a space partitioning tree where the leaf nodes are data objects and each nonleaf node is recursively subdivided into four quadrants. This paper deploys an in-memory quadtree to implement fast coarse-grained spatial range queries. We keep an ID interval $[StartID, EndID]$ for each inner node representing the descendants' ID range. Considering that there is a one-one mapping between

a quadrant of the quadtree and a partition of the Z curve, the objects satisfying the Z order can be obtained by recursively accessing the four quadrants of the quadtree.

Example 2: For the dataset in Table 1, we build a quadtree, as shown in Figure 3. The third child of the root corresponds to the southwest partition of the Z curve in the left part of Figure 3. Since the third child of the root has three nonempty leaf nodes (filled in black) with IDs 3, 4, and 5, the ID interval of this node is [3, 5].

ARM-SQ BASED BRQ ALGORITHM

Based on ARM-SQ, an efficient ARMSQ-BRQ algorithm is proposed in this paper. ARMSQ-BRQ adopts a generation-test framework. In the generation stage, we generate candidates; in the test stage, we test the candidates and obtain the ultimate answer. The major procedures of ARMSQ-BRQ include three major steps.

Spatial Query on Quadtree

As suggested in Christoforaki et al. (2011), a fine-grained spatial query deteriorates the query performance. In this paper, we also deploy a coarse-grained spatial query for $q.R$. Specifically, we check whether $q.R$ is fully contained in a sub-quadrant of the quadtree in a depth manner until we reach a depth d, where $q.R$ is not fully contained in any sub-quadrants at depth d. Then, we directly return the ID intervals of those sub-quadrants at depth d intersecting $q.R$ as the results of the coarse-grained spatial query. We denote the set of ID intervals as IVs.

Example 3: For a spatial query $q.R = \left\{\left[0,1\right],\left[2,3\right]\right\}$ shown as the bold rectangle in the left part of Figure 3, as $q.R$ is fully contained in the root of the quadtree, we recursively check it in each of the four child nodes of the root. Then, we find that $q.R$ is not fully contained in any of these four child nodes, so we return the ID intervals [0,1] for the first child of the root and [3,5] for the third child of the root because these two nodes intersect with $q.R$. As a result, $IVs = \left\{\left[0,1\right],\left[3,5\right]\right\}$.

FIS Selection and Keyword Query on the Materialized Inverted Index

FIS Selection

The main challenge facing ARMSQ-BRQ is that given a query $q = \left\langle q.R, q.W \right\rangle$ and a set of mined FISs F, we can choose the appropriate FISs for $q.W$ to answer the query.

To tackle this challenge, the frequent subsets of $q.W$ are defined as follows:

Definition 2: The frequent subset of $q.W$: a subset of $q.W$ containing all elements in $q.W$ that appear in at least one FIS of F.

We denote $q.W_f$ as the frequent subset of $q.W$. Correspondingly, $q.W_n = q.W - q.W_f$ is denoted as the infrequent subset of $q.W$. As the inverted lists corresponding to the elements in $q.W_n$ will not be materialized and each of these lists can be accessed directly, we focus on the accessing strategy for $q.W_f$ in the later description. Then, the definition of an FIS covered by $q.W_f$ is given as follows:

Definition 3: An FIS covered by $q.W_f$: an FIS f is covered by $q.W_f$ if $f \subseteq q.W_f$.

Correspondingly, all FISs covered by $q.W_f$ are denoted as $F^{q.W_f}$.

Example 4: For $q.W = \{a,b,c,e\}$ and the mined FISs $F = \{\{a\},\{b\},\{c\},\{a,b\},\{b,c\},\{d\}\}$, we first split $q.W$ into 2 subsets using Definition 2, as element e does not appear in any FIS of F, so we obtain $q.W_f = \{a,b,c\}$ and $q.W_n = \{e\}$. Then, we can obtain $F^{q.W_f} = \{\{a\},\{b\},\{c\},\{a,b\},\{b,c\}\}$ using Definition 3.

As $F^{q.W_f}$ may contain multiple FISs, choosing the appropriate p FISs $\{f_1, f_2, ..., f_p\}$ satisfying the following 3 constraints is the key to this paper:

1. $\bigcup_{i\in[1,p]} f_i = q.W_f$
2. $\forall i \neq j,\ f_i \cap f_j = \varnothing$
3. The total query cost is minimized.

Considering that the above problem is essentially a set covering problem with NP complexity, we design a benefit-based greedy FIS selection algorithm, BGF-Selection, to calculate an approximate optimal solution. Here, the benefit of f is defined as follows:

Definition 4: The benefit of f: given an FIS f, its benefit is the reduced costs by materializing f.

We denote the benefit of f as B_f. It is difficult to precisely calculate B_f, as it depends on the IVs obtained from spatial query, the specific list intersection algorithm, etc. To simplify the calculation, we approximately compute B_f according to Equation (4) as follows:

$$B_f = \sum_{e\in f} |S_e| - |S_f| \tag{4}$$

where $\sum_{e\in f} |S_e|$ is the sum of supports for all elements in f, and $|S_f|$ is denoted as the support corresponding to FIS f.

Based on the analyses above, BGF-Selection can be implemented as follows:

1. Set $F^S = \varnothing$;
2. Sort the FISs in $F^{q.W_f}$ in descending order of benefit;
3. For each $f \in F^{q.W_f}$, if $f \cap f' = \varnothing$ for all $f' \in F^S$, then $F^S = F^S \cup \{f\}$.

To efficiently check whether f intersects with any FIS in F^S, a flag array with a length $|q.W_f|$ is used, in which each element of $q.W_f$ corresponds to a unique position in the array. For the current accessing FIS f, if the corresponding flag of element $e \in f$ in the array has been set, then f must intersect with at least one FIS in F^S; otherwise, we set the flags for all elements in f and append f into F^S.

Example 5: We compute B_f according to Definition 4 for each f in $F^{q.W_f}$. For FIS $\{a,b\}$, as the supports for $\{a\}$, $\{b\}$, and $\{a,b\}$ are 4, 4, and 3, respectively, we can compute $B_{\{a,b\}} = 4+4-3 = 5$. Similarly, we compute the benefits for FIS $\{a\}$, $\{b\}$, $\{c\}$, $\{b,c\}$ as 0, 0, 0, 4, respectively.

We then sort $F^{q.W_f}$ in benefit descending order and obtain $F^{q.W_f} = \{\{a,b\},\{b,c\},\{a\},\{b\},\{c\}\}$. For the 1st FIS $\{a,b\}$ in $F^{q.W_f}$, we add it directly into F^S. For FIS $\{b,c\}$, as it intersects with FIS $\{a,b\}$ in F^S, we ignore this FIS. We scan each FIS in $F^{q.W_f}$ in a similar way, and finally, we obtain $F^S = \{\{a,b\},\{c\}\}$.

Keyword Query on Materialized Inverted Index

After FISs are selected, we execute a keyword query on the inverted index corresponding to F^S and $q.W_n$; then, we store the candidates of the query in C. We use the same DAAT list intersection strategy as in SFC-Quad, so we do not discuss it in detail.

Example 6: For the $IVs = \{[0, 1], [3, 5]\}$ obtained in the spatial query, we query them in the inverted lists pointed by $\{a,b\}$, $\{c\}$ and $\{e\}$. As IDs 0 and 1 appear in all 3 lists, these 2 IDs are stored in C as our candidates.

Candidate Verification

As we use a coarse-grained spatial query in the spatial query stage, the candidates obtained in C may not satisfy the spatial constraint. Therefore, we need a verification step here to verify each candidate in C to check whether the object of this candidate is in $q.R$.

Example 7: For candidates 0 and 1 in example 6, we check their coordinates and find that the coordinates (0, 0) of object 0 are not in $q.R$, so the final result is object 1.

Based on the discussion above, Figure 4 gives the final ARMSQ-BRQ algorithm.

Analysis: Assuming that there m mined 1-FISs (only one element in the FIS), in the worst case, ARM-SQ generates $\sum_{i=2}^{h} C(m,i)$ FISs whose number of elements is between 2 and h. Therefore, the additional space overhead for materialization is $l * \sum_{i=2}^{h} C(m,i)$, where l is the average length of the materialized inverted lists and $C(m,i)$ is a combination number.

The major time overhead for ARMSQ-BRQ is accessing elements in the inverted index. After materialization, we can decrease the number of lists to be accessed from $|q.W| = |q.W_f| + |q.W_n|$ to $|F^S| + |q.W_n|$, where $|F^S| \leq |q.W_f|$. For each list, we access at most $\sum_{i=1}^{|IV|} |IVs[i]|$ elements, where $IVs[i]$ is the i-th ID interval in IVs and $|IVs[i]|$ is the length of $IV[i]$. Therefore, the maximum number of elements that need to be accessed is $(|F^S| + |q.W_n|) * \sum_{i=1}^{|IV|} |IVs[i]|$.

Figure 4. The ARMSQ-BRQ Algorithm

Algorithm: ARMSQ-BRQ

Input: q: a query with the form $<q.R, q.W>$
Output: RS: spatial keyword query result
1. $IVs \leftarrow$ a collection of ID intervals by retrieving $q.R$ in quadtree
2. split $q.W$ into $q.W_f$ and $q.W_n$ according to Definition 2
3. $F^{q.W_f} \leftarrow$ All FISs covered by $q.W_f$ according to Definition 3
4. $F^S \leftarrow$ BGF-Selection($F^{q.W_f}$)
5. $C \leftarrow$ the IDs in IVs which appeared in all inverted lists pointed by each FIS in F^S and element in $q.W_n$
6. $RS \leftarrow$ verified ids in C whose object is in $q.R$

EXPERIMENT

All experiments were conducted on a PC with an Intel i7-11800H CPU running at 2.3 GHz CPU and 16 GB RAM running Windows 10. All algorithms were implemented in Java with jdk1.8, and IntelliJ IDEA 2021.1.3 x64 was used as the compiler.

Datasets

We use ROAD (http://www.diag.uniroma1.it//challenge9/download.shtml) and EURO (https://www.pocketgpsworld.com/) as the spatial datasets and RETAIL and KOSARAK (http://fimi.uantwerpen.be/data) as the keyword datasets. ROAD is the road network dataset of the United States with a total of 20 million coordinates. EURO is a European point-of-interest dataset with 100,000 coordinates. RETAIL is a sales dataset of a retail store in Belgium, with a total of 88,162 records. KOSARAK is the clickstream dataset of a Hungarian online news portal with a total of 990,000 records. We show the details of the 2 keyword datasets in Table 3. We combine EURO and RETAIL into a new spatial keyword dataset (DS1 in short) by randomly and non-repeatedly assigning a coordinate in EURO to each record in RETAIL. Similarly, KOSARAK and ROAD are composed of a dataset DS2 of 990,000 records.

Parameter Settings

In this paper, ARM-SQ is evaluated with the other 6 indexes: IF-R* (Zhou et al., 2005), WIBR-Tree (Wu et al., 2012b), IR-Tree (Wu et al., 2012a), CIBR (Wu et al., 2012b), SFC-Quad (Christoforaki et al., 2011), and KR*-Tree (Hariharan et al., 2007). Each of these indexes may involve some specific parameters. We experimentally choose the parameter value that gives the corresponding index the best performance. For the R-tree and R*-tree-based algorithms, page_size is set to 32K, and fanout

Table 3. Statistics on Keyword Dataset

Property	RETAIL	KOSARAK
Total number of objects	88162	990000
Total number of words	90875	8019013
Total number of unique words	16469	41270
Average length of inverted list	6	194

is set to 4000. As WIBR-Tree iteratively divides all objects in the dataset into two groups using the most frequent words, we set the number of frequent words to 10 and 20 for DS1 and DS2, respectively. For CIBR, we set the number of clusters to 20 and 40 for DS1 and DS2, respectively.

For ARM-SQ, the support thresholds are 400 and 5000 for DS1 and DS2, respectively. Next, we evaluate the effects of materialization depth h, the number of high-frequency words x in a query, the number of total words y in a query, and the query range percentage p (the ratio of query rectangle over the entire region of the corresponding dataset).

The Effects of h

To evaluate the effects of h on ARM-SQ, we fix x =4, y =6, p =10% and alter h with 2, 3, 4, 5, and 6 accordingly, then randomly choose 1000 objects for each h from the corresponding dataset as queries. The results on space cost, query cost, and total cost on the two datasets are shown in Figures 5 and 6, respectively.

Figure 5. The Effects of h on DS1

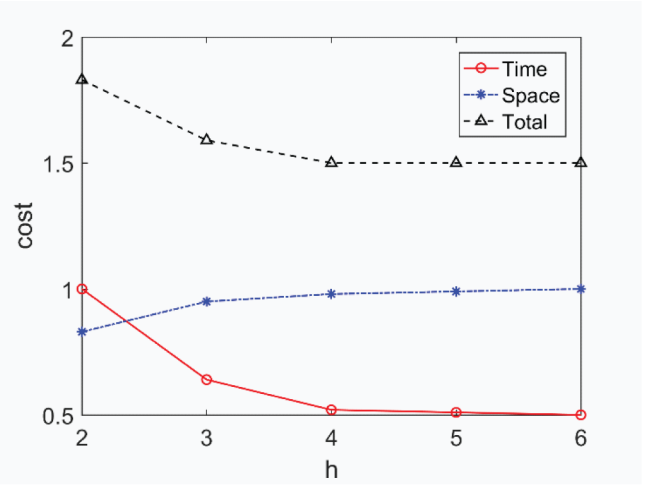
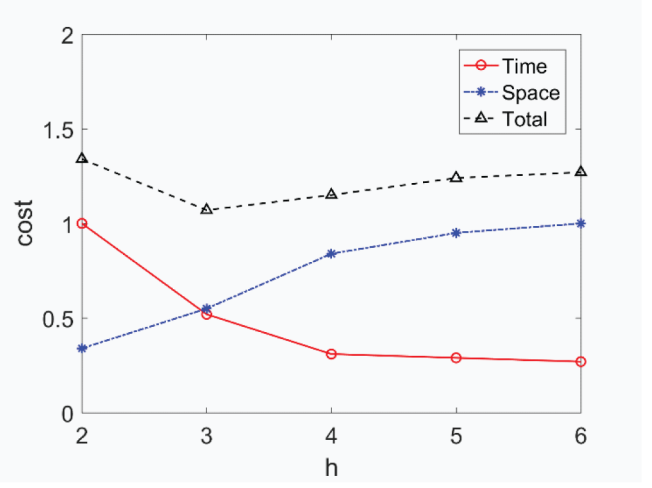


Figure 6. The Effects of h on DS2



As seen from Figures 5 and 6, the space cost gradually increases with the increase of h, while the query time has exactly the opposite trends. DS1 is relatively small, so the total cost tends to be a straight line when h is not less than four. The total cost is the smallest when h is three on DS2. We observe that a small h is good enough to achieve a good balance between time and space. For simplicity, h =4 is used as the default materialization depth in the following experiments unless otherwise specified. In specific applications, we can adjust h according to the actual preference for space and time.

The Effects of x

To evaluate the effects of x on ARM-SQ, we fix y =6, p =10% and alter x with 1, 2, 3, 4, 5, and 6 accordingly, then randomly choose 1000 objects for each x from the corresponding dataset as queries. The results on the two datasets are shown in Figures 7 and 8, respectively.

Figure 7. The Effects of x on DS1

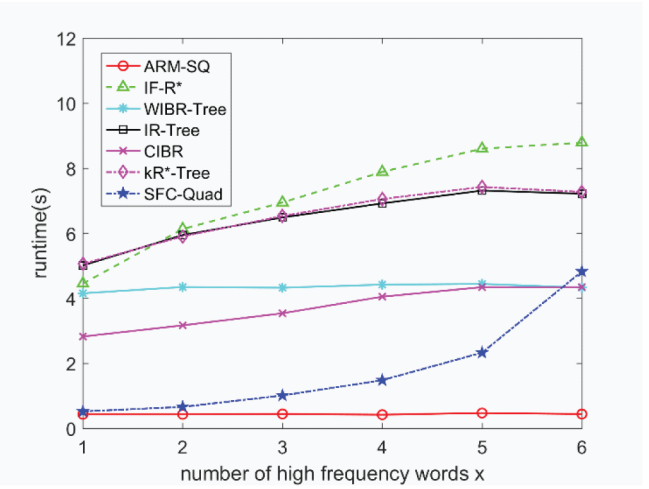
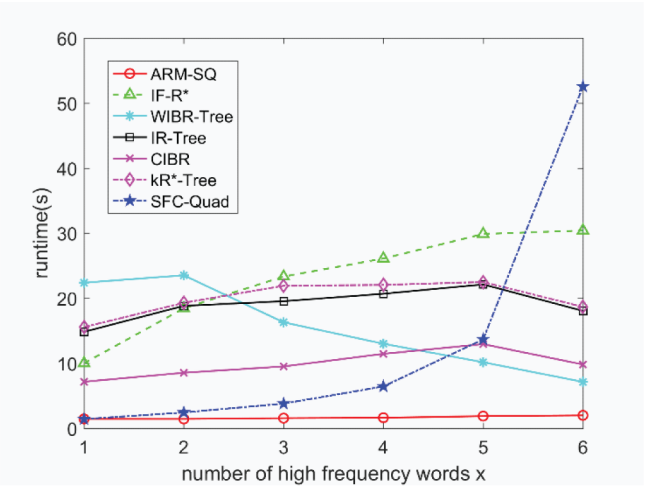


Figure 8. The Effects of x on DS2



Figures 7 and 8 show that the query efficiency of ARM-SQ is much better than that of the other competitors. The performance improvements of ARM-SQ over SFC-Quad on the two datasets can be over 11 and 26 times, respectively. The reason why ARM-SQ significantly outperforms SFC-Quad lies in the substantially decreased disk accesses. The average disk accesses of ARM-SQ in DS2 are 128, 136, 165, 160, 338 and 213 for x varying from 1 to 6, respectively, whereas these numbers for SFC-Quad are 293, 657, 1113, 1875, 3806, and 8546. The query times of ARM-SQ are almost fixed with the increase of x, which shows that our materialization strategy can significantly reduce the intersection cost of long lists. Except for ARM-SQ and WIBR, the query times of the other algorithms increase significantly with the increase of x in most cases. The curves of SFC-Quad jump sharply when x is not less than five, showing that SFC-Quad suffers more from a larger x. The curve of WIBR-tree drops rapidly on DS2 because of a relatively high skewness on the KOSARAK dataset. Despite this, the efficiency of WIBR-tree is much lower than ARM-SQ.

The Effects of y

To evaluate the effects of y on ARM-SQ, we fix x =4, p =10% and alter y with 4, 6, 8, and 10 accordingly, then randomly choose 1000 objects for each y from the corresponding dataset as queries. The results on the two datasets are shown in Figures 9 and 10, respectively.

It can be seen from Figure 9 and 10 that with the increase in y, the proportion of frequent words gradually decreases. Nevertheless, ARM-SQ's query efficiency is superior to the competitors. The performance improvements of ARM-SQ over SFC-Quad on the two datasets can be over 10 and 26 times, respectively. Similarly, the average disk accesses of ARM-SQ in DS2 are 45, 160, 205, and 228 for y, varying from four to ten, respectively, whereas these numbers for SFC-Quad are 7652, 1875, 1274, and 1063. As a result, the query performances can be greatly improved. The query times of ARM-SQ, SFC-Quad, and CIBR decrease with increasing y because they process infrequent words first, and more infrequent words lead to a smaller intersection result.

The Effects of p

To evaluate the effects of p on ARM-SQ, we fix x =4, y =6 and alter p by 1%, 5%, 10%, 25%, and 50% then randomly choose 1000 objects for each p from the corresponding dataset as queries. The results on the two datasets are shown in Figures 11 and 12, respectively.

Figure 9. The Effects of y on DS1

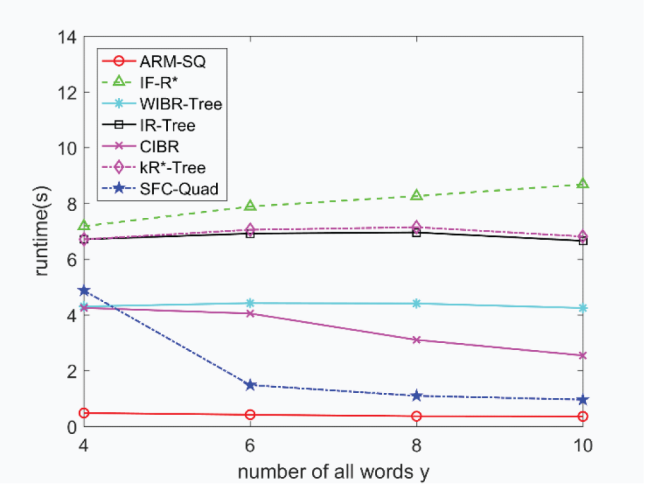


Figure 10. The Effects of y on DS2

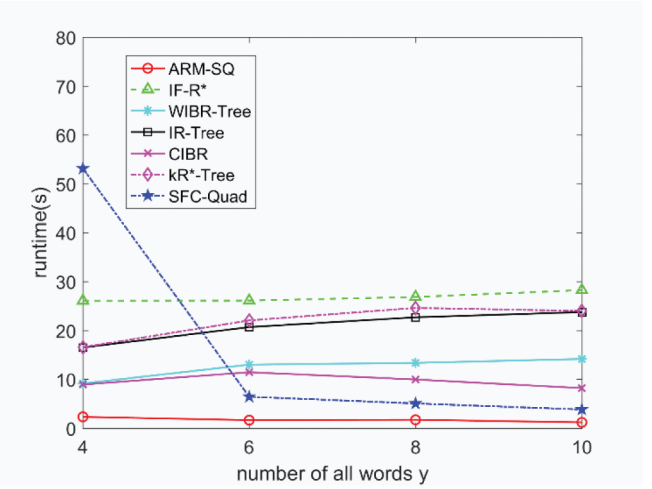
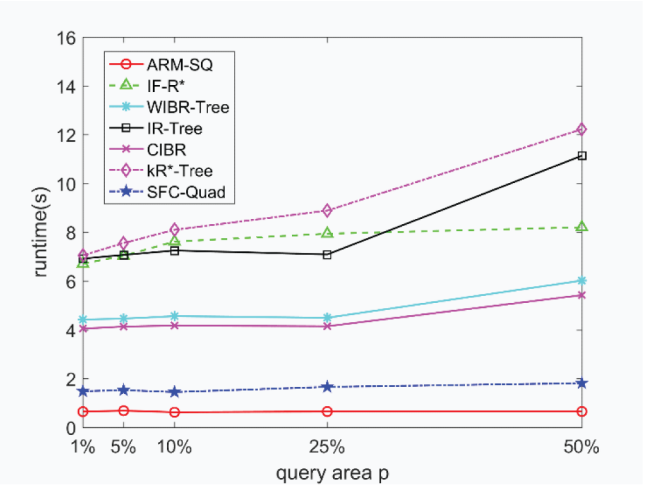


Figure 11. The Effects of p on DS1

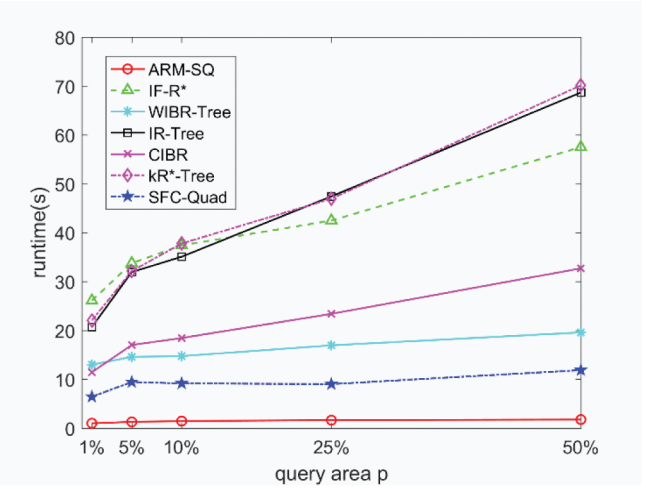


We can see from Figures 11 and 12 that the query times of all indexes increase with increasing p. When p increases from 1% to 50%, the query times on DS2 of IF-R*, WIBR, IR, CIBR, KR*, SFC-Quad, and ARM-SQ increase by 120%, 50%, 231%, 185%, 218%, 84%, and 60%, respectively. In this case, the increase ratio of ARM-SQ remains low and is only slightly higher than WIBR-tree, which indicates that it has good scalability with the increase of p. The increase ratio of ARM-SQ is lower than that of SFC-Quad. The reason lies in that ARM-SQ materializes multiple long lists into a smaller list, so the accessed list elements are smaller than those of SFC-Quad.

CONCLUSION

Given the deficiency that most existing algorithms do not consider the affects of frequent keywords, this paper proposes a novel association rule mining-based spatial keyword index ARM-SQ and

Figure 12. The Effects of p on DS2



the corresponding spatial keyword query algorithm ARMSQ-BRQ. Experimental results show the superiority of our structure and algorithm. In addition, our index and algorithm have the following good characteristics:

1. Our materialization strategy can be applied to any text-first spatial index, improving the performance of many existing applications.
2. ARMSQ-BRQ works well for both scenarios, with and without numerous high-frequency words, showing better adaptability to word frequency distribution.

One weakness of ARM-SQ lies in the offline mining process introduced by ARM, which may introduce a higher index updating cost. We can alleviate this problem by rebuilding the indexes periodically when the server is not busy. Future research should include studies of other efficient materialization methods. In addition, extending ARM-SQ to trajectory data, privacy protection, and many other scenarios will be fascinating directions.

COMPETING INTERESTS

The authors of this publication declare there is no conflict of interest.

FUNDING INFORMATION

This research was supported by the National Natural Science Foundation of China [grant numbers 62262035, 62262034, 62062046, 11961141001, and U1831204].

REFERENCES

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conference on Very Large Databases, VLDB*. ACM.

Alghyaline, S., Hsieh, J.-W., & Lai, J. Z. (2016). Efficiently mining frequent itemsets in transactional databases. *Journal of Marine Science and Technology*, *24*(2), 12. doi:10.6119/JMST-015-0709-1

Chen, L., Cong, G., Jensen, C. S., & Wu, D. (2013). Spatial keyword query processing: An experimental evaluation. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *6*(3), 217–228. doi:10.14778/2535569.2448955

Chen, L., Shang, S., Yang, C., & Li, J. (2020). Spatial keyword search: A survey. *GeoInformatica*, *24*(1), 85–106. doi:10.1007/s10707-019-00373-y

Chen, Z., Chen, L., Cong, G., & Jensen, C. S. (2021). Location-and keyword-based querying of geo-textual data: A survey. *The VLDB Journal*, *30*(4), 603–640. doi:10.1007/s00778-021-00661-w

Chen, Z., Cong, G., Zhang, Z., Fuz, T. Z. J., & Chen, L. (2017). Distributed publish/subscribe query processing on the spatio-textual data stream. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE. doi:10.1109/ICDE.2017.154

Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., & Suel, T. (2011). Text vs. space:efficient geo-search query processing. *CIKM '11: Proceedings of the 20th ACM International Conference on Information and knowledge Management*, (pp. 423–432). ACM. doi:10.1145/2063576.2063641

Cui, N., Li, J., Yang, X., Wang, B., Reynolds, M., & Xiang, Y. (2019). When geo-text meets security: Privacy-preserving boolean spatial keyword queries. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, (pp. 1046-1057). IEEE. doi:10.1109/ICDE.2019.00097

Ding, J., Li, H., Deng, B., Jia, L., & You, J. (2022). Fast mining algorithm of frequent itemset based on spark [in Chinese]. *Journal of Software*. doi:10.13328/j.cnki.jos.006404

Gao, C., & Jensen, C. S. (2016). Querying geo-textual data: spatial keyword queries and beyond. *SIGMOD '16: Proceedings of the 2016 International Conference on Management of Data*, (pp. 2207-2212). ACM. doi:10.1145/2882903.2912572

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Record*, *29*(2), 1–12. doi:10.1145/335191.335372

Han, Y., Wang, L., Zhang, Y., Zhang, W., & Lin, X. (2015). Spatial keyword range search on trajectories. In M. Renz, C. Shahabi, X. Zhou, & M. Cheema (Eds.), Lecture Notes in Computer Science: Vol. 9050. *Database Systems for Advanced Applications. DASFAA 2015*. Springer. doi:10.1007/978-3-319-18123-3_14

Hariharan, R., Hore, B., Li, C., & Mehrotra, S. (2007). Processing spatial-keyword (SK) queries in geographic information retrieval (GIR) systems. *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, (pp. 16-16). IEEE. doi:10.1109/SSDBM.2007.22

Jia, L., Kong, M., Wang, W., Li, M., & You, J. (2021). 2D Hilbert encoding and decoding algorithms on skewed data (in Chinese). *Journal of Tshinghua University (Science and Technology)*. 10.16511/j.cnki.qhdxxb.2021.21.043

Lee, T., Park, J., Lee, S., Hwang, S., Elnikety, S., & He, Y. (2015). Processing and optimizing main memory spatial-keyword queries. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *9*(3), 132–143. doi:10.14778/2850583.2850588

Mahmood, A. R., Aly, A. M., & Aref, W. G. (2018). FAST: frequency-aware indexing for spatio-textual data streams. *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, (pp. 305–316) IEEE.

Qian, Z., Xu, J., Zheng, K., Zhao, P., & Zhou, X. (2018). Semantic-aware top-k spatial keyword queries. *World Wide Web (Bussum)*, *21*(3), 573–594. doi:10.1007/s11280-017-0472-y

Salgado, C., Cheema, M. A., & Ali, M. E. (2018). Continuous monitoring of range spatial keyword query over moving objects. *World Wide Web (Bussum)*, *21*(3), 687–712. doi:10.1007/s11280-017-0488-3

Su, S., Teng, Y., Cheng, X., Xiao, K., Li, G., & Chen, J. (2015). Privacy-preserving top-k spatial keyword queries in untrusted cloud environments. *IEEE Transactions on Services Computing*, *11*(5), 796–809. doi:10.1109/TSC.2015.2481900

Sun, J., Xu, J., Zheng, K., & Liu, C. (2017). Interactive spatial keyword querying with semantics. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, (pp. 1727-1736). ACM. doi:10.1145/3132847.3132969

Tampakis, P., Spyrellis, D., Doulkeridis, C., Pelekis, N., Kalyvas, C., & Vlachou, A. (2021). A novel indexing method for spatial-keyword range queries. *17th International Symposium on Spatial and Temporal Databases*. ACM. doi:10.1145/3469830.3470897

Wang, X., Ma, J., Li, F., Liu, X., Miao, Y., & Deng, R. H. (2021). Enabling efficient spatial keyword queries on encrypted data with strong security guarantees. *IEEE Transactions on Information Forensics and Security*, *16*, 4909–4923. doi:10.1109/TIFS.2021.3118880

Wang, X., Ma, J., Liu, X., Deng, R. H., Miao, Y., Zhu, D., & Ma, Z. (2020). Search me in the dark: Privacy-preserving boolean range query over encrypted spatial data. *IEEE Conference on Computer Communications*, (pp. 2253-2262). IEEE. doi:10.1109/INFOCOM41043.2020.9155505

Wu, D., Cong, G., & Jensen, C. S. (2012a). A framework for efficient spatial web object retrieval. *VLDB Journal - The International Journal on Very Large Data Bases, 21*(6), 797-822. 10.1007/s00778-012-0271-0

Wu, D., Man, L. Y., Cong, G., & Jensen, C. S. (2012b). Joint Top-K Spatial Keyword Query Processing. *IEEE Transactions on Knowledge and Data Engineering*, *24*(10), 1889–1903. doi:10.1109/TKDE.2011.172

Yan, H., Ding, S., & Suel, T. (2009). Inverted index compression and query processing with optimized document ordering. *Proceedings of the 18th International Conference on World Wide Web*, (pp. 401-410). ACM. doi:10.1145/1526709.1526764

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, *12*(3), 372–390. doi:10.1109/69.846291

Zhang, C., Tian, P., Zhang, X., Liao, Q., Jiang, Z. L., & Wang, X. (2019). HashEclat: An efficient frequent itemset algorithm. *International Journal of Machine Learning and Cybernetics*, *10*(11), 3003–3016. doi:10.1007/s13042-018-00918-x

Zhang, R., Chen, W., Hsu, T.-C., Yang, H., & Chung, Y.-C. (2019). ANG: A combination of Apriori and graph computing techniques for frequent itemsets mining. *The Journal of Supercomputing*, *75*(2), 646–661. doi:10.1007/s11227-017-2049-z

Zhao, S., Cheng, X., Su, S., & Shuang, K. (2017). Popularity-aware collective keyword queries in road networks. *GeoInformatica*, *21*(3), 485–518. doi:10.1007/s10707-017-0299-9

Zheng, B., Zheng, K., Xiao, X., Su, H., Yin, H., Zhou, X., & Li, G. (2016). Keyword-aware continuous knn query on road networks. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE.

Zhou, Y., Xie, X., Wang, C., Gong, Y., & Ma, W. Y. (2005). Hybrid index structures for location-based web search. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, (pp. 155-162). ACM. doi:10.1145/1099554.1099584

Zhu, X., & Liu, Y. (2019). An efficient frequent pattern mining algorithm using a highly compressed prefix tree. *Intelligent Data Analysis*, *23*(S1), 153–173. doi:10.3233/IDA-192645

Lianyin Jia is an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. He Received his Ph.D. degree in Computer Science from South China University of Technology, Guangzhou, China in 2013. His current research interests include database, data mining, information retrieval and parallel computing.

Haotian Tang is an MPhil student in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. His current research interests include database, data ming, parallel computing.

Mengjuan Li is a Librarian in the Department of Technology, Library, Yunnan Normal University, Kunming, China. She received her master degree in Computer Science from Kunming University of Science and Technology, Kunming, China in 2008. Her current research interests include information retrieval, parallel computing.

Binxin Zhao is an MPhil student in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. His current research interests include database, data mining.

Shoulin Wei is an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology. He received his Ph.D. from University of Chinese Academy of Sciences in 2017. His current research interests include cloud computing, data mining.

Haihe Zhou is a Lecturer in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. He received his Master degree in Computer Science from Kunming University of Science and Technology, Kunming, China in 2003. His current research interests include database, data mining.