Collaborative Spatial Keyword Top-k Query

Liang Liu*, Shuai Guo*, Xiaolin Qin*, Qinxue Wang*
* College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China
liangliu@nuaa.edu.cn, marvel_agent@nuaa.edu.cn, qinxcs@nuaa.edu.cn, nuaawqx@163.com

***Abstract*—The proliferation of geo-social network, such as Foursquare and Flickr, enables users to generate location information and its corresponding descriptive keywords. Spatial keyword queries are used to find objects in the geo-social networks. Typical spatial keyword queries meet only a single user's need at a time, which contain a single query location and a single set of query keywords. Collaborative Spatial Keyword Top-k Query (TKCSKQ) asks for top-k objects that are close to multiple query positions and theirs keywords have high relevancy with multi-group query keywords. To solve the problem that there are repeated and synonymous keywords in multi-group query keywords, a keywords similarity calculation formula based on the weight of query keywords weight is designed. And we propose SKNIR-tree to support near keywords matching, which is an extension of the IR-tree. Based on the SKNIR-tree, we propose a query processing algorithm that prunes search space through maintaining a priority queue and calculating the minimum spatial and textual similarity of each node with the query, to quickly identify the desired objects. Extensive experiments on real dataset validate the efficiency and the scalability of the proposed algorithm.**

Keywords: spatio-textual object, spatial keyword query, Top-k query, collaborative query

I. INTRODUCTION

With the wide application of the localization technology, geotagging is incorporated into the text data. For example, photo sharing sites (e.g., Flickr) have many photos which contain the location and text description. As another example, check-ins or reviews in location based social networks (such as Foursquare) contain both text description and locations of points of interest. Spatial keyword query processing technologies [1,2,3,4] are used to identify the desired spatio-textual objects efficiently, which have high relevancy with the query while taking into account both the spatial proximity and the text similarity.

Typical spatial keyword queries meet only a single user's need at a time, which take a single query location and a single set of keywords as input parameters, return the objects that theirs locations are near the query point and theirs keywords are highly similar to the query keywords. But in real life, users often draw up a plan collaboratively. The query in these applications is submitted by multiple users. For example, users who are in different companies dine together. Each user wants the restaurant to be near its own location and the restaurant's description is similar to its own need. In this paper, we study how to find suitable top-k objects to meet multiple users' needs. We formulate a new kind of query called collaborative spatial keyword top-k query (TKCSKQ), which aims to retrieve top-k objects for meeting multiple users' needs. Compared with traditional spatial keyword queries, TKCSKQ faces the following challenges:

(1)There are repeated and synonymous keywords in query keywords submitted by multiple users. Traditional spatial keyword queries take a single set of keywords submitted by a single user as parameter, and there are no duplicate keywords or near keywords in the query keywords. Theirs keywords similarity calculation method does not consider the weight of the query keywords.

(2)There is mismatch problem because of the synonymous keywords. For example, in a collaborative query, two users propose a query keyword "open-air" and "outdoor" respectively. Clearly, they both want to query outdoor restaurant. For an object that contains "open-air" keyword, it will only match one query keyword while using traditional query processing technology. But in fact, it should match the two query keywords.

(3)How can we process TKCSKQ efficiently? It is another great challenge for TKCSKQ to quickly find top-k objects that are close to multiple query points and theirs keywords have high relevancy with query keywords.

To solve the above problem, we proposes a collaborative spatial keyword top-k query processing technique, and the main contributions are as follows:

(1)To solve the problem that multiple users submit the repeated or synonymous keywords, we design the keywords similarity calculation formula based on the weight of query keywords.

(2)To process TKCSKQ efficiently and solve mismatch problem, we propose an efficient hybrid index structure called Synonymous Keywords Normalization IR-tree (SKNIR-tree), which normalizes all the keywords and uses NKI (Normalized Keyword Identification) to represent keyword, to maximize the users' satisfactions.

(3)Based on the SKNIR-tree, we propose an algorithm TKCSK (Top-K Collaborative Spatial Keyword processing method) that prunes search space through maintaining a priority queue and calculating the minimum spatial and textual similarity of each node with the query, to quickly identify the desired objects.

In order to evaluate the performance of the SKNIR-tree and TKCSK algorithm, we conduct extensive experiments in two aspects of query time and IO. The results demonstrate that the proposed algorithm is efficient and scalable and exhibits superior performance over the brute force method.

The rest of this paper is organized as follows. Section II introduces the related work. We formally define the problem of collaborative spatial keyword top-k query in

Section III. Section IV introduces the SKNIR-tree. Section V introduces our algorithm for processing TKCSKQ. Section VI presents our experimental evaluation. We summarize our work and discuss future work in Section VII.

II. RELATED WORK

Typical spatial keyword queries meet only a single user's need at a time [5,6,7,8]. They mainly construct the hybrid index and propose the corresponding algorithm to search desired objects [9,10,11]. The R*-IF [9] organizes location information with R tree, each leaf node is associated with an inverted file to organize text information. The algorithm finds the nearest neighbor according to the leaf nodes of R tree. And then in each leaf node, the objects are sorted according to the textual relevancy. IR-tree [2] associates an inverted file with each node of the R tree, and uses the priority queue to query objects with the maximum relevancy taking into account both the spatial proximity and keywords relevancy. BR-tree [10] organizes text information through associating a bitmap with each node of R tree. The algorithm prunes search space according to whether the bitmap contains all query keywords. Then the objects are sorted according to the distance. Wu et al. [11] study the authentication of moving top-k spatial keyword queries using the MIR-tree, which modifies the IR-tree by embedding a series of digests in each node of the tree. The above queries are all submitted by a single user. On the contrary, we aim to solve the spatial keyword queries submitted by multiple users and find the desired objects to meet the needs of multiple users.

The existing spatial keyword queries have some collaborative studies, ALI et al. [12] studies the k-BEST-SUBGROUPS-NN query. The query is submitted by multiple users, and asks for results to meet any sub-groups' demand. The algorithm proposes a data centric approach, gradually accesses the objects from centric, and identifies the best subset at each step. The main idea is to develop the best subset of the visited objects by moving the query point radially from the centroid, without enumerating all possible subsets. But this paper only considers the coordination of space, does not consider keywords. Zhang et al. [13] proposes TkCoS query taking into account both the spatial proximity and keywords relevancy, and designs the STR-tree which prunes search space by calculating the upper boundary and the lower boundary for each node set. TkCoS query is submitted by multiple users, and finds the top-k object sets to satisfy the users' needs. The collective spatial keyword queries [14,15,16] are submitted by a single user, and find the top-k object sets. The keywords of each object set contain query keywords, the location of the object set is close to the query location. TKCSKQ is different from the above researches, the query helps multiple users in different locations to identify top-k objects collaboratively while taking into account the problem of the repeated and synonymous keywords.

III. PROBLEM STATEMENT

Spatial textual object. $o =< \rho, \varphi >$, where ρ is the object's location, φ is a set of keywords of the object.

Collaborative Spatial Keyword Top-k Query. $Q = \{< q_1.\rho, q_1.\varphi >, \dots, < q_n.\rho, q_n.\varphi >\}$, where $q_i.\rho$ is the ith user's query location, $q_i.\varphi$ is the ith user's query keywords. TKCSKQ asks for top-k objects that are close to multiple users' locations and theirs texts are highly similar to the query keywords.

In order to find the best top-k objects from the dataset, we propose a ranking function to measure how well an object satisfies TKCSKQ, as shown in Formula 1. The function takes into account both the spatial proximity and keywords relevancy. In Formula 1, $\alpha \in [0,1]$ is the user preference on spatial proximity and keywords relevancy. The spatial proximity, denoted by $D(Q, o)$, is obtained by the maximum distance between q_i and object(shown in Formula 2). $maxD$ denotes the maximal distance between any two objects in dataset. It is used as a normalization factor. In Formula 3, $TRel(Q.\varphi, o.\varphi)$ is the keywords relevancy. Traditional spatial keyword queries are submitted by a single user, there are no repeated and synonymous keywords in query keywords. So their keywords similarity calculation formula does not consider the weight of a single query keyword. But for TKCSKQ, multiple users may submit the same keywords or synonyms keywords, thus query keywords need to be assigned different weights. We propose a keyword similarity calculation formula based on the weight of query keywords (shown in Formula 3). If t_i represents the keyword that both appear in the query keywords and object keywords, w_i is the weight of t_i, obtained by the number of times t_i's NKI appears in the query keywords. The smaller the value calculated by Formula 1, the more satisfied the query condition.

Finally, the goal of a TKCSKQ is to find top-k objects with the smallest $S_{sk}(\mathrm{Q}, \mathrm{o})$. Our problem can be defined as Definition 1.

$$S_{sk}(\mathrm{Q}, \mathrm{o}) = \alpha \frac{D(Q,o)}{maxD} + (1-\alpha)(1 - TRel(Q.\varphi, o.\varphi)) \quad (1)$$

$$D(Q, o) = \max_{1 \le i \le n}(\mathrm{dist}(q_i.\rho, \mathrm{o}.\rho)) \quad (2)$$

$$TRel(Q.\varphi, o.\varphi) = \frac{w_1 + w_2 + \cdots + w_i}{num(Q.\varphi)} \quad (3)$$

Definition 1 (*TkCSKQ Retrieval*) Given a dataset and a TKCSKQ, find top-k objects $\{o_1, o_2, \dots, o_k\}$, such that there does not exist o′that satisfies $\mathrm{o}' \notin \{o_1, o_2, \dots, o_k\}$ and $S_{sk}(\mathrm{Q}, \mathrm{o}') < S_{sk}(\mathrm{Q}, o_i)$, $o_i \in \{o_1, o_2, \dots, o_k\}$.

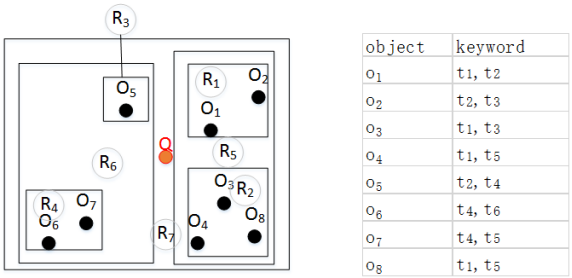
IV. SKNIR-TREE

Figure 1 is an example of eight spatial textual objects. The left shows the locations of the objects. And the right shows the keywords information, among them, keyword t_2 and t_5 are semantically synonymous.

To answer TKCSKQ efficiently, we introduce an efficient hybrid index structure called SKNIR-tree, which is an extension of IR-tree, as shown in figure 2. It can efficiently standardize the nonstandard keyword into the NKI, to ensure the accuracy and efficiency of the query.

SKNIR-tree normalizes the keywords by maintaining a relational table (shown at the bottom left of Figure 2) and

identifies it with an integer number called NKI. The synonymous keywords will be translated into the same NKI. The application scenarios (e.g., multiple users who are in different corporates dine) TKCSKQ process involve the keywords such as fast food, open-air restaurants and other tabbed keywords, and the number of tabbed keywords is always fixed. Therefore, NKI can be determined in advance, and it is easier to correspond the



object	keyword
o_1	t_1, t_2
o_2	t_2, t_3
o_3	t_1, t_3
o_4	t_1, t_5
o_5	t_2, t_4
o_6	t_4, t_6
o_7	t_4, t_5
o_8	t_1, t_5

Fig.1 Spatial textual objects

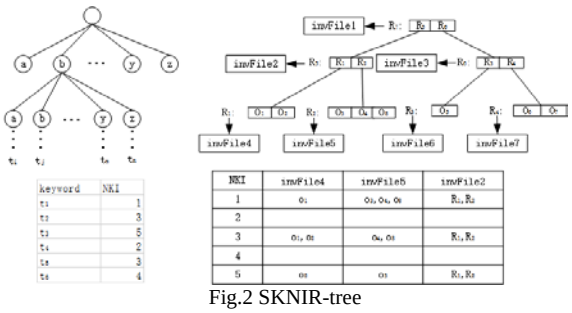


Fig.2 SKNIR-tree

normal keywords to the NKI. Although it is laborious, but only need to be done once. To speed up the query of keywords in the relational table, we use trie (shown at the left above of Figure 2) to organize nonstandard keywords. In the trie, we store an English letter in each node in addition to the root node, each keyword can be accessed through a unique path following its letter order. Each leaf node's form in trie is (*key*, *P*), where *key* is a keyword, and *P* is a pointer to the keyword in the relational table. During the query process, we start from the root node in trie, then in the root node's sub-nodes, hash technology is used to query the node location of the first letter of the keyword, until finding the last letter. The trie uses hash technology to store and query the location of the nodes, thus the query efficiency is high.

We did a proper transformation based on IR-tree to fit TKCSKQ (shown at the right of Figure 2). In the inverted file, NKI replaces the original object keyword. Each leaf node contains entries of the form (*op*, *o.r*, *IFp*), where *op* is the pointer to the object o, *o.r* is the bounding rectangle of o, and *IFp* is the pointer to the inverted file. The inverted file contains two main components: first, all distinct NKIs appearing in the corresponding objects; second, posting lists for each NKI nki that is a sequence of identifiers of the objects whose NKIs contain nki. Each non leaf node contains entries of the form (*nps*, *r*, *IFp*), where *nps* is the pointer to the child nodes, *r* is the minimum bounding rectangle of all rectangles in entries of the child node, and *IFp* is the pointer to the inverted file.

Figure 2 gives an example of SKNIR-tree for the objects in Figure 1. In the relational table, the nonstandard keywords are standardized into digital tags, and both t_2 and t_5 are normalized to 3 because they have the same meaning. The R tree is constructed according to the locations of the objects, and the NKI inverted files are constructed for each node.

Here we describe the construction of the SKNIR-tree, as shown in the algorithm 1. The overall NKI is pre created, and then the keywords that appear in the dataset are identified with the corresponding NKI (line 1-5). Then insert the (keyword, NKI) into the relational table (line 7), and insert the nonstandard keywords and their address in the relational table into the trie (line 8). Finally, we call the algorithm Insert in IR-tree [2] to insert the object. It is worth noting that the insertion parameters of the SKNIR-tree are the minimum bounding rectangle of the object and the NKIs (line 10).

Algorithm 1: IndexBuilding(o)

```
1.   for each o
2.     for each keyword t in o
3.       identifying t with NKI
4.     end for
5.   end for
6.   for each nonredundant t
7.     insert (t,NKI) to a relational table
8.     insert t to trie
9.   end for
10.  Insert(MBR,NKIs)
```

V. PROCESSING TKCSKQ

In this section, two Baseline algorithms are first proposed. Then, based on SKNIR-tree, an efficient algorithm for TKCSKQ processing is proposed.

A. *Baseline Algorithms*

Baseline 1 Unite Subquery (US). Traditional spatial keyword queries are submitted by a single user, and return the objects that theirs locations are near the query point and theirs texts are highly similar to the query keywords. TKCSKQ is submitted by multiple users, including multiple query locations and multi-group query keywords, and return the objects that theirs locations are near the multiple query points and theirs texts are highly similar to the multi-group query keywords. Intuitively, a brute force approach is to process each subquery q_i in Q using traditional query processing technology independently, and merge all the results returned by the subqueries. Obviously, this approach will lead to high processing cost. First, the same node will be accessed repeatedly in different subqueries. Second, we need to keep the number of the result of each sub-query sufficiently large, to ensure the merged result contains the top-k.

Baseline 2 First Space Then Text (FSTT). This algorithm uses Formula 3 and relational table to calculate the text relevancy of all objects based on the inverted file. The calculation result is denoted as *TRank*. Then through extending the method of searching neighbor objects [18], the algorithm incrementally finds neighbors that are closest to multiple users using R-tree, and maintains top-k result through calculating neighbors' spatial textual relevancy based on Formula 1. The algorithm keeps track of the maximum text relevancy in *TRank*, denoted by *MaxT* that has not been calculating so far. For a newly calculated object in R-tree, if the combined score

computed from its location and *MaxT* exceeds kth result object, the Algorithm stops since it is guaranteed that all un-calculated objects will not have a lower score than the current kth result object.

B. TKCSK Algorithm

In this section, we propose the TKCSK algorithm to processing TKCSKQ. The algorithm maintains a priority queue that stores minimum spatial textual relevancy between Q and SKNIR-tree nodes. The relevancy calculation function is shown in Formula 4. $\min S_{sk}(\mathrm{Q},\mathrm{N})$ is the relevancy between Q and the node N. And the relevancies between Q and the objects in minimum bounding rectangle of node N are all greater than $\min S_{sk}(\mathrm{Q},\mathrm{N})$. We give the formal definition in Theorem 1, and give the proof. The priority queue is arranged from small to large. The algorithm iterates over the elements from the head, and calculates the $\min S_{sk}(\mathrm{Q},\mathrm{N})$ of its child nodes, then inserts them into the queue. If we get an object from the head of the queue, then the object is the one of the top-k. Until we get the all top-k result, the algorithm stops. Other nodes and objects in the priority queue do not need to be accessed and calculated to achieve the purpose of fast pruning search space and improving query efficiency.

$$\min S_{sk}(\mathrm{Q},\mathrm{N}) = \alpha \frac{D(Q,N)}{maxD} + (1-\alpha)(1 - TRel(Q.\varphi, N.\varphi)) \quad (4)$$

Theorem 1 Given a TKCSKQ Q and a node N of SKNIR-tree, and the minimum bounding rectangle of the node N contains the objects os, then $\forall \mathrm{o} \in \mathrm{os}\ (\min S_{sk}(\mathrm{Q},\mathrm{N}) \leq S_{sk}(\mathrm{Q},\mathrm{o}))$.

Proof. $\mathrm{dist}(q_i.\rho, N) \leq \mathrm{dist}(q_i.\rho, \mathrm{o}.\rho)$, then $\min_{1\leq i\leq n}(\mathrm{dist}(q_i.\rho, N)) \leq \min_{1\leq i\leq n}(\mathrm{dist}(q_i.\rho, \mathrm{o}.\rho))$, $D(Q,N) \leq D(Q,o)$. And because the node N contains all the keywords of the objects os, $1 - TRel(Q.\varphi, N.\varphi) \leq 1 - TRel(Q.\varphi, o.\varphi)$. In summary, $\min S_{sk}(\mathrm{Q},\mathrm{N}) \leq S_{sk}(\mathrm{Q},\mathrm{o})$.

The algorithm's pseudocode is shown in Algorithm 2. First of all, we transform the query keywords from multiple users into NKIs, and sort NKIs (line 2-7). Then we calculate the number of time each NKI appears as the weight of the NKI (line 8). The algorithm maintains a priority queue U which is arranged from small to large according to S_{sk}, and firstly the root node of the SKNIR-tree is stored in U (line 9-10). If U is not empty and the number of result is less than *k* (line 11), the algorithm iteratively checks the first element E in U. If E is an object, it is returned as a top-k result. If E is a leaf node, we compute the $S_{sk}(\mathrm{Q},\mathrm{o})$ of E's objects and push them into U. If E is a non leaf node, we compute the $\min S_{sk}(\mathrm{Q},\mathrm{N})$ of E's child nodes and push them into U. (line 12-23).

Algorithm 2: Search(index R, TKCSKQ Q)

```
1.  Result ← ∅
2.  for each q_i.φ
3.    for each keyword t
4.      Q.NKIs.add(t->NKI)
5.    end for
6.  end for
7.  sort(Q.NKIs)
8.  Q.NKI.w ← count(Q.NKI)
9.  U ← EmptyPriorityQueue
10. U.push(R.root,0)
11. while U is not empty and Result.length<k
12.   E ← U.pop()
13.   if E is an object
14.     result.add(E)
15.   else if E is a leaf node
16.     for each object o in the leaf node
17.       U.push(o, S_sk(Q, o))
18.     end for
19.   else
20.     for each node n in E
21.       U.push(n, minS_sk(Q, N))
22.     end for
23.   end if
24. end while
```

VI. EXPERIMENTS

A. Experimental Setting

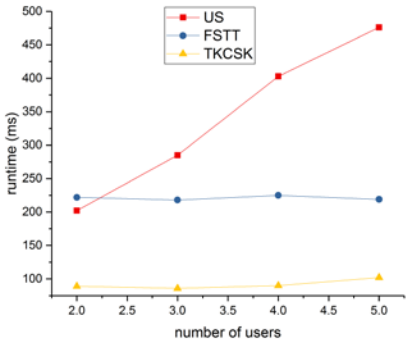
The experiment is performed on ThinkPad T450, with the following configuration: CPU: Intel (R) Core (TM) i5-5200U CPU @ 2.20GHZ, RAM: 6G, Hard disk: 500G, Operation System: Windows 10. All algorithms of the experiment are implemented in Java, and the integrated development environment is IntelliJ IDEA Community Edition 14.0.2.

We use the yelp_academic_dataset_business dataset [17] provided by the Yelp web site in the experiments. It collects 85,901 restaurants from 11 cities in 4 countries. Each line in the dataset records a restaurant's information which contains 31 items, such as merchant identification, address, latitude and longitude, classification etc. We use latitude and longitude as object location and use classification as object keywords. We also extend the dataset by the method of random sampling based on the original dataset. Because the keywords in the dataset do not have synonymous keywords, we randomly select multi-group 2-4 keywords as synonymous keywords.

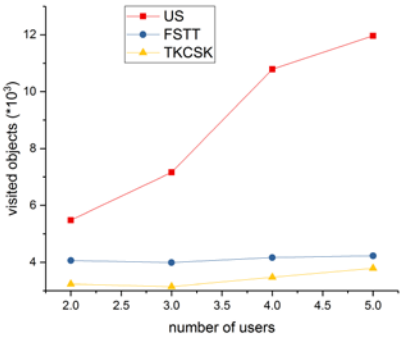
The existing technologies about spatial keyword query cannot deal with TKCSKQ. Thus we only compare our TKCSK algorithm with two Baseline algorithms proposed in the 5.1 section. We normalize the object keywords and then put it into memory in FSTT algorithm in advance.

B. Performance Evaluation

We compare TKCSK algorithm with two Baseline algorithms in two aspects of query efficiency and IO cost, and the IO cost is measured by the number of objects accessed. In the following, *n* is the number of users, *k* is the number of results, and α is the user's preferences on spatial proximity and keywords relevancy.

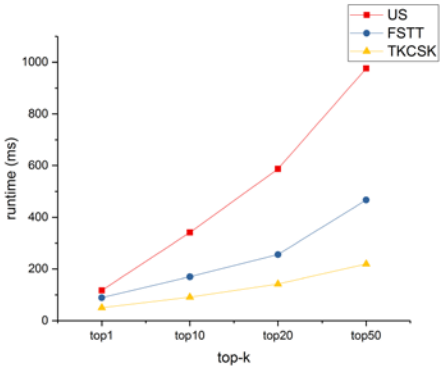


a) query time

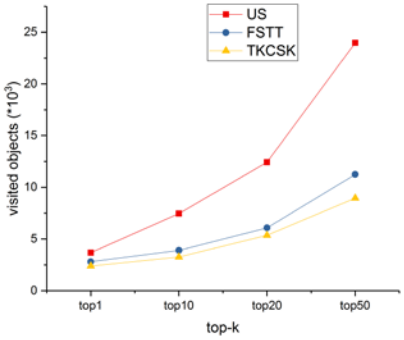


b) IO

Fig.3 Effect of n



a) query time



b) IO

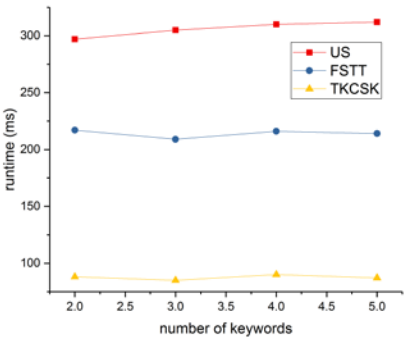
Fig.4 Effect of k

(1) Effect of *n*

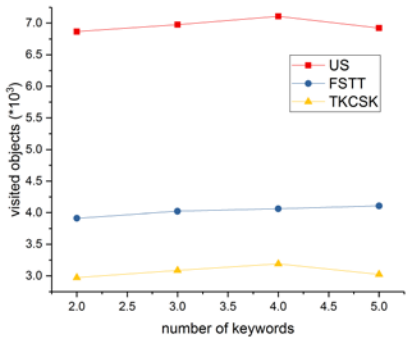
Here, we fix *k* at 10, the number of query keywords for each user at 3 and α at 0.5. Fig.3 shows the impact of different number of users on query time and IO cost. Because each subquery in the US algorithm will access the index once, and each subquery needs to maintain a result that is much larger than k to ensure the fusion result including top-k, so the query time and IO cost of US algorithm are much larger than that of TKCSK algorithm. In the FSTT algorithm, all the object NKIs are stored in memory, and the algorithm incrementally finds neighbors that are closest to multiple users using R-tree. The algorithm stops when the combined score computed from its location and *MaxT* exceeds kth result. So the number of object accessed is not much different from that of TKCSK algorithm as shown in Fig.3(b). However, as each iteration step of the algorithm needs to find the *MaxT* in *TRank*, though it may not be needed every step, but it also causes high cost, so the query time of FSTT algorithm is greater than that of TKCSK. In addition, the number of subqueries increases with the increase of n in US algorithm, so as shown in the figure, both the query time and the IO cost increase. The increase of n does not affect the pruning rate of TKCSK and FSTT algorithms, so as shown in the figure, with the increase of n, the query time and IO cost of TKCSK and FSTT algorithms are almost unchanged.

(2) Effect of *k*

In this set of experiments, we evaluate the performance of the three algorithms with a varying *k* while fixing *n* at 3, the number of query keywords for each user at 3 and α at 0.5. As shown in Fig.4(a) and Fig.4(b), the TKCSK algorithm has shorter query time and smaller IO cost than two Baseline algorithms for all values of k. With k increasing, as the number of results increases, the amount of pruning will decrease accordingly, so as shown in the figure, query time and IO cost increase. And since the US algorithm needs to maintain a larger value than k (experimental setting is 2 times), its growth rate is greater than that of TKCSK algorithm and FSTT algorithm.



a) query time

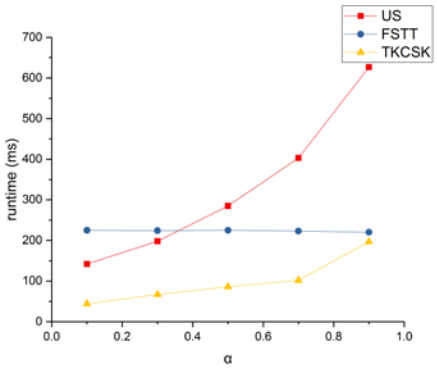


b) IO

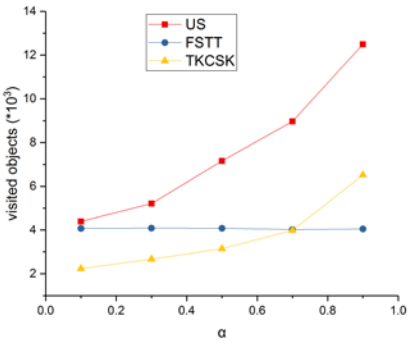
Fig.5 Effect of keywords

(3) Effect of the number of query keywords

Fig.5 shows the effect of the number of query keywords for each user on query time and IO cost when we fix k at 10, n at 3, and α at 0.5. Specifically, TKCSK algorithm has shorter query time and smaller IO cost than two Baseline algorithms for all values of the number of keywords. As shown in figure, query time and IO cost remain unchanged with the number of query keywords increasing, because the number of keyword queries does not affect the pruning rate of the all algorithm.



a) query time

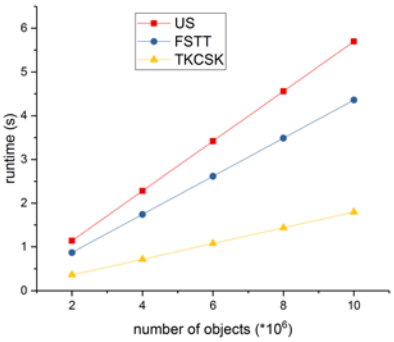


b) IO

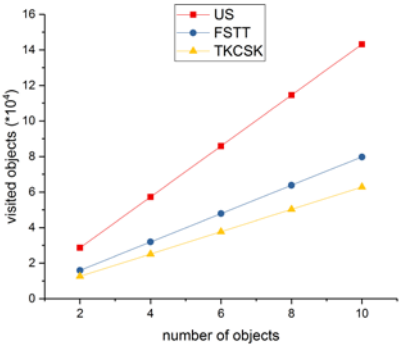
Fig.6 Effect of α

(4) Effect of α

Fig.6 shows the effect of α on query time and IO cost when we fix k at 10, n at 3 and the number of query keywords for each user at 3. Specifically, TKCSK algorithm has shorter query time and smaller IO cost than two Baseline algorithms for all values of the α. Recall that α is used to adjust user's preferences for spatial proximity and keywords relevancy. The greater α value is, the more user cares about the location of results. The smaller alpha value is, the more user cares about the keywords relevancy of results. As shown in the figure, query time and IO cost of US and TKCSK algorithms increase with the increase of α. This is because the spatial differentiation is small and the pruning rate using spatial proximity is small. For FSTT Algorithm, because it firstly uses R tree to incrementally calculate the object closest to multiple users, the change of α will not affect algorithm's termination condition, therefore query time and IO cost of the FSTT algorithm do not change with α increasing.



a) query time



b) IO

Fig.7 Scalability

(5) Scalability

In order to evaluate the scalability of TKCSK, we generate dataset from two million to ten million based on the original dataset. The location of the generated object is the random neighbors of the location of the object in the original dataset, and the keyword is randomly obtained from the keyword set in the original dataset. Fig.7 shows the tendency of query time and IO cost of algorithms with changing the amount of data when we fix k at 10, n at 3, α at 0.5 and the number of query keywords for each user at 3. As shown in the figure, TKCSK algorithm is scalable and better than two Baseline algorithms.

VII. CONCLUSIONS

In this paper, we study the problem of collaborative spatial keyword top-k query (TKCSKQ), which aims to find top-k objects that are close to multiple query points and theirs texts have high relevancy with query keywords. Because there are repeated and synonymous keywords in query keywords, we design the keywords similarity

calculation formula based on the weight of query keywords. To solve mismatch problem and efficiently process TKCSKQ, we present an efficient query processing algorithm that is based on a hybrid index called SKNIR-tree. The algorithm prunes search space through maintaining a priority queue and calculating the minimum spatial and textual similarity of each node with the query locations and query keywords, to quickly identify the desired objects. Our experimental evaluation shows that the proposed algorithm is efficient and scalable and superior performance compared with two baseline methods.

REFERENCES

[1] De Felipe I, Hristidis V, Rishe N. Keyword Search on Spatial Databases[C]// IEEE, International Conference on Data Engineering. IEEE Computer Society, 2008:656-665.

[2] Cong G, Jensen C S, Wu D. Efficient retrieval of the Top-k most relevant spatial web objects[J]. Proceedings of the Vldb Endowment, 2009, 2(1):337-348.

[3] Chen Y Y, Suel T, Markowetz A. Efficient query processing in geographic web search engines[C]// ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, Usa, June. DBLP, 2006:277-288.

[4] Cao X, Cong G, Jensen C S. Retrieving Top-k prestige-based relevant spatial web objects[J]. Proceedings of the Vldb Endowment, 2010, 3(1):373-384.

[5] Cao X, Chen L, Cong G, et al. Spatial Keyword Querying[M]// Conceptual Modeling. Springer Berlin Heidelberg, 2012:16-29.

[6] Chen L, Cong G, Jensen C S, et al. Spatial keyword query processing: an experimental evaluation[J]. Proceedings of the Vldb Endowment, 2013, 6(3):217-228.

[7] De Felipe I, Hristidis V, Rishe N. Keyword Search on Spatial Databases[C]// IEEE, International Conference on Data Engineering. IEEE Computer Society, 2008:656-665.

[8] Zhang D, Tan K L, Tung A K H. Scalable Top-k spatial keyword search[C]// International Conference on Extending Database Technology. ACM, 2013:359-370.

[9] Zhou YH, Xie X, Wang C, GongYC, Ma WY. Hybrid index structures for location-based Web search. In: Proc. of the CIKM. New York: ACM Press, 2005. 155−162. [doi: 10.1145/1099554.1099584].

[10] Zhang DX, Chee YM, Mondal A, Tung AKH, Kitsuregawa M. Keyword search in spatial databases: Towards searching by document. In: Proc. of the ICDE. Washington: IEEE, 2009. 688−699. [doi: 10.1109/icde.2009.77] .

[11] Wu D., Choi B., Xu J., C.S. Jensen. Authentication of moving top-k spatial keyword queries. IEEE Trans. Knowl. Data Eng., 27 (4) (2015), pp. 922-935

[12] Ali M E, Tanin E, Scheuermann P, et al. Spatial Consensus Queries in a Collaborative Environment[J]. Acm Transactions on Spatial Algorithms & Systems, 2016, 2(1):3.

[13] Zhang J, Meng X, Zhou X, et al. Co-spatial Searcher: Efficient Tag-Based Collaborative Spatial Search on Geo-social Network[C]// International Conference on Database Systems for Advanced Applications. Springer-Verlag, 2012:560-575.

[14] Cao X, Cong G, Jensen C S, et al. Collective spatial keyword querying[C]// ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June. DBLP, 2011:373-384.

[15] Zhang P, Lin H, Yao B, et al. Level-aware Collective Spatial Keyword Queries ☆[J]. Information Sciences, 2016, 378(C):194-214.

[16] Long C, Wong C W, Wang K, et al. Collective spatial keyword queries: a distance owner-driven approach[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2013:689-700.

[17] https://www.yelp.ca/dataset_challenge/dataset

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