International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

Saclable Nearest Neighbourhood Keyword Cover Searching Using Keyword-NNE

Tudimella Sowndarya & Ram Mohan Reddy
M.Tech., C.S.E., Department
Email: sowndarya2512@gmail.com
Head of Department,C.S. E., Department
Email:- rammohanreddy51@gmail.com
Newton's Institute of Engineering, Guntur. Andhra Pradesh

Abstract:

It is fundamental that the articles in a spatial database (e.g., diners/lodgings) are connected with keyword(s) to exhibit their associations/organizations/features. An interesting issue known as Closest Keywords look is to address objects, called catchphrase cover, which together cover a game plan of request watchwords and have the base between objects partitioned. Starting late, we watch the growing availability and criticalness of catchphrase rating in dissent appraisal for the better essential administration. This goads us to investigate a non particular type of Closest Keywords look for called Best Keyword Cover which considers between objects discrete and moreover the watchword rating of items.The standard count is impelled by the procedures for Closest Keywords look for which relies upon altogether joining objects from different request catchphrases to deliver confident watchword covers. Exactly when the amount of request watchwords assembles, the execution of the measure count drops radically due to enormous confident watchword covers delivered. To ambush this hindrance, this work proposes an impressively more versatile count called watchword nearest neighbor augmentation (catchphrase NNE). Appeared differently in relation to the example computation, watchword NNE algorithm significantly reduces the amount of contender catchphrase covers made. The all around examination and expansive tests on certifiable enlightening files have legitimized the power of our catchphrase NNE computation.

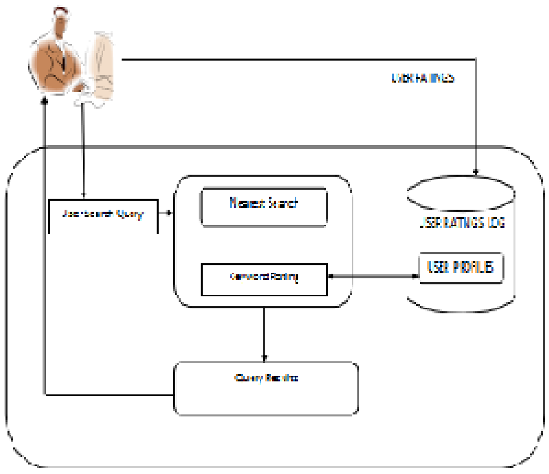
Keywords : Index Terms—Spatial database, point of interests, keywords, keyword rating, keyword cover

Introduction:

DRIVEN by portable registering, area based administrations and wide accessibility of broad advanced maps and satellite symbolism (e.g., Google Maps and Microsoft Virtual Earth benefits), the spatial catchphrases look issue has pulled in much consideration recently]. In a spatial database, each tuple speaks to a spatial question which is related with keyword(s) to show the data, for example, its organizations/administrations/highlights.
Given an arrangement of question catchphrases, a fundamental errand of spatial watchwords seek is to distinguish spatial object(s) which are related with watchwords pertinent to an arrangement of inquiry watchwords, and have attractive spatial connections (e.g., near each other as well as near an inquiry area). This issue has one of a kind incentive in different applications since clients' necessities are frequently communicated as numerous catchphrases. For instance, a traveler who intends to visit a city may have specific shopping, feasting and settlement needs. It is alluring that every one of these necessities can be fulfilled without long separation voyaging.

Because of the amazing an incentive by and by, a few variations of spatial catchphrase thought is to consolidate nodes in higher various leveled levels of KRR*-trees to create applicant catchphrase covers. At that point, the most encouraging hopeful is surveyed in need by consolidating their kid nodes to create new hopefuls.

System Architecture:



Regardless of the way that BKC request can be effectively settled, when the amount of request catchphrases extends, the execution drops radically in light of tremendous cheerful catchphrase covers made. To vanquish this essential drawback, we developed much adaptable catchphrase nearest neighbor advancement (watchword NNE) count which applies a substitute procedure. Catchphrase NNE picks one inquiry catchphrase as focal request watchword. The articles related with the principle question watchword are critical articles. For each central dissent, the adjacent best course of action (known as close-by best catchphrase cover ðlbkcþ) is prepared. Among them, the lbkc with the most dumbfounding evaluation is the game plan of BKC request. Given a central challenge, its lbkc can be perceived by fundamentally recuperating two or three adjoining what's all the more, exceptionally assessed dissents in each non-essential request catchphrase (two-four inquiries in ordinary as outlined in tests).
Contrasted with the gauge calculation, the number of hopeful watchword covers created in catchphrase NNE calculation is fundamentally diminished. The top to bottom investigation uncovers that the quantity of competitor catchphrase covers further prepared in catchphrase NNE calculation is ideal, and every watchword applicant cover preparing produces much less new competitor catchphrase covers than that in the gauge calculation.seek issue have been considered. The works plan to locate various person objects, each of which is near an inquiry area and the related catchphrases (or called archive) are extremely significant to an arrangement of question watchwords (or called question record). The record comparability (e.g., [14]) is connected to gauge the importance between two arrangements of catchphrases. Since it is likely none of individual items is related with all question watchwords, this inspires the investigations to recover numerous objects, called watchword cover, which together cover (i.e., related with) all question catchphrases and are near each other. This issue is known as m Closest Keywords (mCK) question in]. The issue considered in [4] moreover requires the recovered questions near an inquiry area. This paper researches a non specific form of mCK inquiry, called Best Keyword Cover (BKC) question, which considers between objects remove and in addition watchword rating. It is spurred by the perception of expanding accessibility and significance of watchword rating in basic leadership.

A huge number of usinesses/ administrations/ includes the world over have been evaluated by clients through online business audit destinations, for example, Yelp, Citysearch, ZAGAT and Dianping, and so on. For instance, an eatery is appraised 65 out of 100 (ZAGAT.com) and a lodging is evaluated 3.9 out of 5 (hotels.com).

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

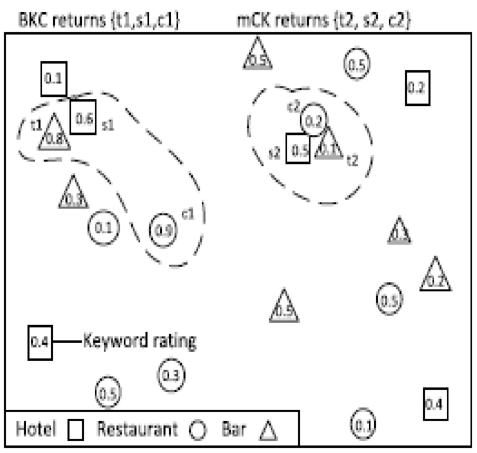


Fig. 1. BKC versus mCK.

thought is to consolidate nodes in higher various leveled levels of KRR*-trees to create applicant catchphrase covers. At that point, the most encouraging hopeful is surveyed in need by consolidating their kid nodes to create new hopefuls.

Despite the fact that BKC inquiry can be successfully settled, when the quantity of inquiry catchphrases expands, the execution drops drastically because of enormous hopeful catchphrase covers created. To defeat this basic downside, we grew much versatile catchphrase closest neighbor development (watchword NNE) calculation which applies an alternate methodology. Catchphrase NNE chooses one question catchphrase as central inquiry watchword. The articles related with the main question watchword are important articles. For every chief protest, the nearby best arrangement (known as nearby best catchphrase cover ðlbkcþ) is processed.

Among them, the lbkc with the most astounding assessment is the arrangement of BKC inquiry. Given a chief protest, its lbkc can be recognized by basically recovering a couple of adjacent what's more, very evaluated protests in each non-important inquiry catchphrase two-four questions in normal as delineated in tests). Contrasted with the gauge calculation, the number of hopeful watchword covers created in catchphrase NNE calculation is fundamentally diminished. The top to bottom investigation uncovers that the quantity of competitor catchphrase covers further prepared in catchphrase NNE calculation is ideal, and every watchword applicant cover preparing produces much ess new competitor catchphrase covers than that in the gauge calculation.

$$diam(O) = \max_{o_i, o_j \in O} dist(o_i, o_j). \tag{1}$$

$$\begin{aligned} O.score &= score(A, B) \\ &= \alpha\left(1 - \frac{A}{max\_dist}\right) + (1-\alpha)\frac{B}{max\_rating}. \\ A &= diam(O). \\ B &= \min_{o \in O}(o.rating), \end{aligned} \tag{2}$$

where B is the base catchphrase rating of articles in O and að0 □ a □ 1þ is an application specific parameter. In case a ¼ 1, the score of O is only controlled by the estimation of O. In this case, BKC request is spoiled to mCK question. If a ¼ 0, the score of O just considers the base catchphrase rating of articles in Q where max dist and max rating are used to institutionalize estimation and catchphrase rating into [0, 1] independently. max dist is the most extraordinary division between any two dissents in the spatial database D, and max rating is the best catchphrase rating of articles.

Existing System

Some current works center around recovering individual protests by determining a question comprising of an inquiry area and an arrangement of inquiry catchphrases (or known as report in some specific circumstance). Each recovered question is related with watchwords applicable to the inquiry catchphrases and is near the inquiry area. The methodologies proposed by Cong et al. what's more, Li et al. utilize a half and half file that expands nodes in non-leaf nodes of a R/R*-tree with upset lists.

Praposed System

This paper explores a bland form of mCK inquiry, called Best Keyword Cover (BKC) question, which considers between objects separate and additionally catchphrase rating. It is propelled by the perception of expanding accessibility and significance of watchword rating in basic leadership.

A large number of organizations/administrations/includes the world over have been appraised by clients through online business audit destinations, for example, Yelp, Citysearch, ZAGAT and Dianping, and so forth.

Lemma 1. The score is of monotone property.

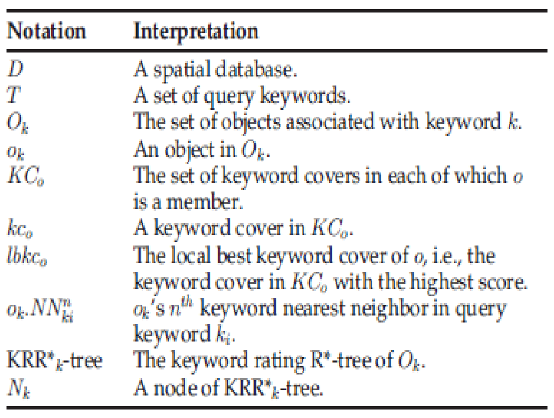
Proof. Given a set of objects O_i, suppose O_j is a subset of O_i. The diameter of O_i must be not less than that of O_j, and the minimum keyword rating of objects in Oi must be not greater than that of objects in O_j. Therefore,Oi:score $\leq O_j$:score.

Definition 2 (Keyword Cover). Let T be a set of keywords {k_1; . . . ; kng and O a set of objects fo1; . . . ; ong, O is a keyword cover of T if one object in O is associated with one and only one keyword in T.

Definition 3 (Best Keyword Cover Query). Given a spatial database D and a set of query keywords T, BKC query returns a keyword cover O of T (O D) such that O:score O0:score for any keyword cover O0 of T (O0 D).

The notations used in this work are summarized in Table 1.

TABLE 1
Summary of Notations



Notation	Interpretation
D	A spatial database.
T	A set of query keywords.
O_k	The set of objects associated with keyword k.
o_k	An object in O_k.
KC_o	The set of keyword covers in each of which o is a member.
kc_o	A keyword cover in KC_o.
$lbkc_o$	The local best keyword cover of o, i.e., the keyword cover in KC_o with the highest score.
$o_k.NN_{ki}^n$	o_k's n^{th} keyword nearest neighbor in query keyword k_i.
KRR*$_k$-tree	The keyword rating R*-tree of O_k.
N_k	A node of KRR*$_k$-tree.

KEYWORD NEAREST NEIGHBOR EXPANSION

Utilizing the baseline calculation, BKC inquiry can be viably settled. Be that as it may, it depends on thoroughly joining objects (or their MBRs). Despite the fact that pruning systems have been investigated, it has been watched that the execution drops significantly, when the quantity of inquiry watchwords expands, in light of the quick increment of competitor catchphrase covers produced. This rouses us to build up an alternate calculation called watchword closest neighbor development. We center around a specific question watchword, called important inquiry catchphrase. The articles related with the foremost question catchphrase are called vital items. Give k a chance to be the foremost inquiry watchword. The arrangement of rule objects is indicated asO_k.

Definition 4 (Local Best Keyword Cover). Given a set of query keywords T and the principal query keyword k 2 T, thelocal best keyword cover of a principal object o_k is

$$lbkc_{ok} = \Big\{ kc_{ok} \mid kc_{ok} \in KC_{ok}, \quad kc_{ok}.score = \max_{kc \in KC_{ok}} kc.score \Big\}, \tag{5}$$

3 RELATED WORK

3.1 Spatial Keyword Search

Recently, the spatial keyword search has received considerable attention from research community. Some existing works focus on retrieving individual objects by specifying a query consisting of a query location and a set of query keywords (or known as document in some context). Each retrieved object is associated with keywords relevant to the query keywords and is close to the query location [3], [5], [7], [8], [10], [15], [16]. The similarity between documents (e.g., [14]) are applied to measure the relevance between two sets of keywords. Since it is likely no individual object is associated with all query keywords, some other works aim to retrieve multiple objects which together cover all query keywords [4], [17], [18]. While potentially a large number of object combinations satisfy this requirement, the research problem is that the retrieved objects must have desirable spatial relationship.

In [4], authors put forward the problem to retrieve objects which 1) cover all query keywords, 2) have minimum inter-objects distance and 3) are close to a query location. The work [17], [18] study a similar problem called m Closet Keywords (mCK). mCK aims to find objects which cover all query keywords and have the minimum inter-objects distance. Since no query location is asked in mCK, the search space in mCK is not constrained by the query location. The problem studied in this paper is a generic version of mCK query by also considering keyword rating of objects.

Access Methods

The approaches proposed by Cong et al. [5] and Li et al. [10] employ a hybrid index that augments nodes in non-leaf nodes of an R/R*-tree with inverted indexes. The inverted index at each node refers to a pseudo-document that represents the keywords under the node. Therefore, in order to verify if a node is relevant to a set of query keywords, the inverted index is accessed at each node to evaluate the matching between the query keywords and the pseudo-document associated with the node. In [18], bR*-tree was proposed where a bitmap is kept for each node instead of pseudo-document. Each bit corresponds to a keyword. If a bit is “1”, it indicates some object(s) under the node is associated with the corresponding keyword; “0” otherwise.

A bR*-tree example is shown in Fig. 2a where a non-leaf node N has four child nodes N1, N2, N3, and N4. The bitmaps of N1;N2;N4 are 111 and the bitmap of N3 is 101. In specific, the bitmap 101 indicates some objects under N3 are associated with keyword “hotel” and “restaurant” respectively, and no object under N3 is associated with keyword “bar”. The bitmap allows to combine nodes to generate candidate keyword covers. If a node contains all query keywords, this node is a candidate keyword cover. If multiple nodes together cover all query keywords, they constitute a candidate keyword cover. Suppose the query keywords are 111. When N is visited, its child node N1;N2;N3;N4 are processed. N1;N2;N4 are associated with all query keywords and N3 is associatedwith two query keywords. The candidate keyword covers generated are fN1g, fN2g, fN4g, fN1;N2g, fN1;N3g, fN1;N4g, fN2;N3g, fN2;N4g, fN3;N4g, fN1;N2;N3g, fN1;N3;N4g and fN2;N3;N4g. Among the candidate keyword covers, the one with the best evaluation is processed by combining their child nodes to generate more candidates. However, the number of candidates generated can be very large. Thus, the depth-first bR*-tree browsing strategy is applied in order to access the objects in leaf nodes as soon as possible. The purpose is to obtain the current best solution as soon as possible. The current best solution is used to prune the candidate keyword covers. In the same way, the remaining candidates are processed and the current best solution is updated once a better solution is identified. When all candidates have been pruned, the current best solution is returned tomCK query. In [17], a virtual bR*-tree based method is introduced to handle mCK query with attempt to handle data set with massive number of keywords. Compared to the method in [18], a different index structure is utilized. In virtual bR*-tree based method, an R*-tree is used to index locations of objects and an inverted index is used to label the leaf nodes in the R*-tree associated with each keyword. Since only leaf nodes have keyword information the mCK query is processed by browsing index bottom-up. At first, m inverted lists corresponding to the query keywords are retrieved, and fetch all objects from the same leaf node to construct a virtual node in memory. Clearly, it has a counterpart in the original R*-tree. Each time a virtual node is constructed, it will be treated as a subtree which is browsed in the same way as in [18]. Compared to bR*-tree, the number of nodes in R*-tree has been greatly reduced such that the I/O cost is saved. As opposed to employing a single

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

R*-tree embedded with keyword information, multiple R*-trees have been used to process multiway spatial join (MWSJ) which involves data of different keywords (or types). Given a number of R*-trees, one for each keyword, the MWSJ technique of Papadias et al. [13] (later extended by Mamoulis and Papadias [11]) uses the synchronous R*-tree approach [2] and the window reduction (WR) approach [12]. Given two R*-tree indexed relations, SRT performs two-way spatial join via synchronous traversal of the two R*-trees based on the property that if two intermediate R*-tree nodes do not satisfy the spatial join predicate, then the MBRs below them will not satisfy the spatial predicate either. WR uses window queries to identify spatial regions which may contribute to MWSJ results.

INDEXING KEYWORD RATINGS

To process BKC query, we augment R*-tree with one additional dimension to index keyword ratings. Keyword rating dimension and spatial dimension are inherently different measures with different ranges. It is necessary to make adjustment. In this work, a three-dimensional R*-tree called keyword rating R*-tree (KRR*-tree) is used. The ranges of both spatial and keyword rating dimensions are normalized into [0, 1]. Suppose we need construct a KRR*-tree over a set of objects D. Each object o 2 D is mapped into a new space using the following mapping function:

$$f : o(x, y, rating) \rightarrow o\left(\frac{x}{max_x}, \frac{y}{max_y}, \frac{rating}{max\_rating}\right), \quad (3)$$

where maxx;maxy; max rating are the maximum value of objects in D on x, y and keyword rating dimensions respectively. In the new space, KRR*-tree can be constructed in the same way as constructing a conventional three-dimensional R*-tree. Each node N in KRR*-tree is defined as Nðx; y; r; lx; ly; lrÞ where x is the value of N in x axle close to the origin, i.e., (0, 0, 0, 0, 0, 0), and lx is the width of N in x axle, so do y, ly and r, lr. The Fig. 2b gives an example to illustrate thenodes of KRR*-tree indexing the objects in keyword “restaurant”. In [17], [18], a single tree structure is used to index objects of different keywords. In the similar way as discussed above, the single tree can be extended with an additional dimension to index keyword rating.

A solitary tree structure suits the circumstance that most watchwords are inquiry catchphrases. For the previously mentioned illustration, all catchphrases, i.e., "inn", "eatery" and "bar", are question watchwords. In any case, it is more regular that lone a little portion of watchwords are question catchphrases. For instance in the trials, just under 5 percent catchphrases are question watchwords. In this circumstance, a solitary tree is poor to inexact the spatial connection between objects of couple of particular watchwords. Consequently, numerous KRR*-trees are utilized as a part of this work, each for one keyword.1 The KRR*-tree for watchword ki is signified as KRR*ki-tree. Given a question, the rating of a related catchphrase is regularly the mean of appraisals given by various clients for a timeframe. The change happens yet gradually. Despite the fact that emotional change happens, the KRR*-tree is refreshed in the standard method for R*-tree refresh.

BASELINE ALGORITHM

The baseline algorithm is inspired by the mCK query processing methods [17], [18]. For mCK query processing, the method in [18] browses index in top-down manner while the method in [17] does bottom-up. Given the same hierarchical index structure, the top-down browsing manner typically performs better than the bottom-up since the search in lower hierarchical levels is always guided by the search result in the higher hierarchical levels. However, the significant advantage of the method in [17] over the method in [18] has been reported. This is because of the different index structures applied. Both of them use a single tree

structure to index data objects of different keywords. But the number of nodes of the index in [17] has been greatly reduced to save I/O cost by keeping keyword information with inverted index separately. Since
only leaf nodes and their keyword information are maintained in the inverted index, the bottom-up index browsing manner is used. When designing the baseline algorithm for BKC query processing, we take the advantages of both methods [17], [18]. First, we apply multiple KRR*-trees which contain no keyword information in nodes such that the number of nodes of the index is not more than that of the index in [17]; second, the top-down index browsing method can be applied since each keyword has own index. Suppose KRR*-trees, each for one keyword, have been constructed. Given a set of query keywords T ¼ fk1; . . . ; kng, the child nodes of the root of KRR*ki-tree (i i n) are retrieved and they are combined to generate candidate keyword covers. Given a candidate keyword cover O ¼ fNk1; . . .;Nkng where Nki is a node of KRR*ki-tree.

$$
\begin{aligned}
O.score &= score(A, B).\\
A &= \max_{N_i,N_j \in O} dist(N_i, N_j) \qquad (4)\\
B &= \min_{N \in O}(N.maxrating),
\end{aligned}
$$

where N:maxrating is the maximum value of objects under N in keyword rating dimension; distðNi;NjÞ is the minimum euclidean distance between Ni and Nj in the twodimensional geographical space defined by x and y dimensions.

Lemma 2. Given two keyword covers O and O0, O0 consists of objects fok1; . . . ; okng and O consists of nodes fNk1; . . .;Nkng. If oki is under Nki in KRR*ki-tree for 1 i n, it is true that O0:score O:score. Algorithm 1 shows the pseudo-code of the baseline algorithm. Given a set of query keywords T, it first generates candidate keyword covers using Generate Candidate function which combines the child nodes of the roots of KRR*ki-trees for all ki 2 T (line 2). These candidates are maintained in a heap H. Then, the candidate with the highest score in H is selected and its child nodes are combined using Generate Candidate function to generate more candidates.
Since the number of candidates can be very large, the depth-first KRR*ki-tree browsing strategy is applied to access the leaf nodes as soon as possible (line 6). The first candidate consisting of objects (not nodes of KRR*-tree) is
1. If the total number of objects associated with a keyword is very small, no index is needed for this keyword and these objects are simply processed one by one. the current best solution, denoted as bkc, which is an intermediate solution. According to Lemma 2, the candidates in H are pruned if they have score less than bkc:score (line 8). The remaining candidates are processed in the same way and bkc is updated if the better intermediate solution is found. Once no candidate is remained in H, the algorithm terminates by returning current bkc to BKC query.

Algorithm 1. *Baseline*$(T, Root)$

```
Input: A set of query keywords $T$, the root nodes of all
       KRR*-trees $Root$.
Output: Best Keyword Cover.
1: $bkc \leftarrow \emptyset$;
2: $H \leftarrow Generate\_Candidate(T, Root, bkc)$;
3: while $H$ is not empty do
4:     $can \leftarrow$ the candidate in $H$ with the highest score;
5:     Remove $can$ from $H$;
6:     $Depth\_First\_Tree\_Browsing(H, T, can, bkc)$;
7:     foreach $candidate \in H$ do
8:         if ($candidate.score \leq bkc.score$) then
9:             remove $candidate$ from $H$;
10: return $bkc$;
```

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

Algorithm 2. *Depth_First_Tree_Browsing* (H, T, can, bkc)

```
Input: A set of query keywords $T$, a candidate $can$, the
       candidate set $H$, and the current best solution $bkc$.
1: if can consists of leaf nodes then
2:     $S \leftarrow$ objects in $can$;
3:     $bkc' \leftarrow$ the keyword cover with the highest score
       identified in $S$;
4:     if $bkc.score < bkc'.score$ then
5:         $bkc \leftarrow bkc'$;
6: else
7:     $New\_Cans \leftarrow Generate\_Candidate(T, can, bkc)$;
8:     Replace $can$ by $New\_Cans$ in $H$;
9:     $can \leftarrow$ the candidate in $New\_Cans$ with the
       highest score;
10:    $Depth\_First\_Tree\_Browsing(H, T, can, bkc)$;
```

In Generate Candidate function, it is unnecessary to actually generate all possible keyword covers of input nodes (or objects). In practice, the keyword covers are generated by incrementally combining individual nodes (or objects). An example in Fig. 3 shows all possible combinations of input nodes incrementally generated bottom up. There are three keywords k1; k2 and k3 and each keyword has two nodes. Due to the monotonic property in Lemma 1, the idea of Apriori algorithm [1] can be applied. Initially, each node is a combination with score¼1. The combination with the highest score is always processed in priority to combine one more input node in order to cover a keyword, which is not covered yet. If a combination has score less than bkc:score, any superset of it must have score less than bkc:score. Thus, it is unnecessary to generate the superset. For example, if fN2$_{k2}$;N2$_{k3g}$:score < bkc:score, any superset of fN2$_{k2}$;N2$_{k3g}$ must has score less than bkc:score. So, it is not necessary to generate fN2$_{k2}$; N2$_{k3}$; N1$_{k1g}$ and fN2$_{k2}$; N2$_{k3}$; N2$_{k1g}$.

Algorithm 3. *Generate_Candidate*(T, can, bkc)

```
Input: A set of query keywords $T$, a candidate $can$,
       the current best solution $bkc$.
Output: A set of new candidates.
1: $New\_Cans \leftarrow \emptyset$;
2: $COM \leftarrow$ combining child nodes of $can$ to generate
   keyword covers;
3: foreach $com \in COM$ do
4:     if $com.score > bkc.score$ then
5:         $New\_Cans \leftarrow com$;
6: return $New\_Cans$;
```

KEYWORD NEAREST NEIGHBOR EXPANSION

Using the baseline algorithm, BKC query can be effectively resolved. However, it is based on exhaustively combining objects (oword Cover). Given a set of query keywo$_r$ds T and the principal query keyword k ε T, the local best keyword cover of a principal object ok is

$$lbkc_{ok} = \Big\{ kc_{ok} \mid kc_{ok} \in KC_{ok}, \quad kc_{ok}.score = \max_{kc \in KC_{ok}} kc.score \Big\}, \tag{5}$$

where KC$_{ok}$ is the set of keyword covers in each of which theprincipal object ok is a member. For each principal object ok 2 Ok, lbkc$_{ok}$ is identified. Among all principal objects, the lbkcok with the highest score is called global best keyword cover (GBKC$_k$).

Keyword-NNE

The high performance of keyword-NNE algorithm is due to that each principal node (or object) only retrieves a few keyword-NNs in each non-principal query keyword. Suppose all retrieved keyword-NNs in keyword-NNE algorithm are kept in a set S. In Fig. 13a, the average size of S is shown. The data sets are randomly sampled so that the number of objects in each query keyword in a BKC query is from 100 to 3,000. It illustrates that the impact of the number of objects in query keywords to the size of S is limited. On the contrary, it shows that the size of S is clearly influenced by m. When m increases from 2 to 9, S increases linearly. In average, a principal node (or object) only retrieves 2-4 keyword-NNs in each non-principal query keyword. Fig. 13b shows the number of lbkcs computed in query processing.We can see less than 10 percent of principal nodes (or objects) need to compute their lbkcs in different sizes of data sets. In other words, 90 percent of the overall principal nodes (or objects) are pruned during the query processing.

Weighted Average of Keyword Ratings

The tests compare the weighted average of keyword rating and the minimum keyword rating to performance. The average experimental results of 100 BKC queries on each of four data sets are reported in Fig. 14. We can see the difference between these two situations is trivial. This is because the score computation in the situation of the minimum keyword rating is fundamentally equivalent to that in the situation of weight average. In the former situation, if a combination O of objects (or their MBRs) does not cover a keyword, the rating of this keyword used for computing O:score is 0 while it is the maximum rating of this keyword in the latter situation.

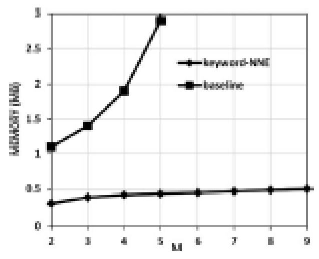
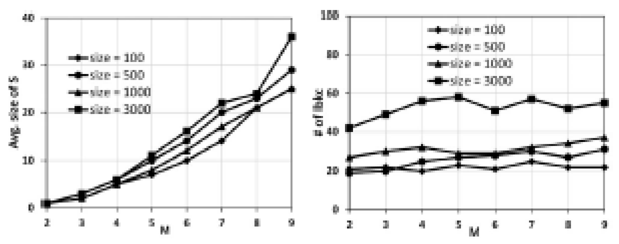


Fig. Maximum memory consumed versus m (a ¼ 0.4).

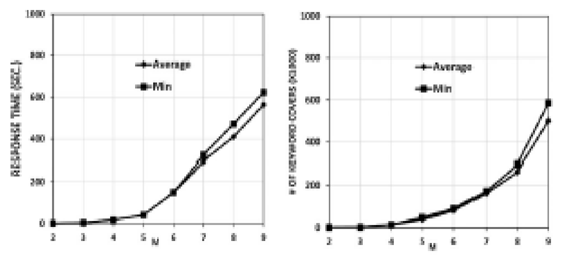
Conclusion

Contrasted with the most significant mCK question, BKC inquiry gives an extra measurement to help more sensible basic leadership. The presented benchmark calculation is motivated by the techniques for handling mCK question. The standard calculation produces an extensive number of applicant catchphrase covers which prompts sensational execution drop when more question watchwords are given. The proposed catchphrase NNE calculation applies an alternate preparing procedure, i.e., scanning nearby best answer for each protest in a specific inquiry watchword. As a result, the quantity of competitor catchphrase covers produced is essentially decreased. The examination uncovers that the quantity of competitor watchword covers which should be additionally prepared in



(a) Average size of *S* vs. *m*. (b) Number of *lbkcs* computed vs. *m*.

Fig. Features of keyword-NNE (a = 0.4).



(a) Response time vs. *m*. (b) Number of nodes visited vs. *m*.

Fig. 14. Weighted average versus minimum (a = 0:4Þ)

keyword NNE calculation is ideal and preparing every watchword competitor cover regularly creates significantly less new applicant catchphrase covers in catchphrase NNE calculation than in the benchmark calculation..

References

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp. 487–499.
[2] T. Brinkhoff, H. Kriegel, and B. Seeger, “Efficient processing of spatial joins using r-trees,” in Proc. ACM SIGMOD Int. Conf. Manage.Data, 1993, pp. 237–246.
[3] X. Cao, G. Cong, and C. Jensen, “Retrieving top-k prestige-based relevant spatial web objects,” Proc. VLDB Endowment, vol. 3, nos.1/2, pp. 373–384, Sep. 2010.
[4] X. Cao, G. Cong, C. Jensen, and B. Ooi, “Collective spatial keyword querying,” in Proc. ACM SIGMOD Int. Conf. Manage. Data,2011, pp. 373–384.
[5] G. Cong, C. Jensen, and D. Wu, “Efficient retrieval of the top-k most relevant spatial web objects,” Proc. VLDB

Endowment, vol. 2, no. 1, pp. 337–348, Aug. 2009.[6] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," J. Comput. Syst. Sci., vol. 66, pp. 614–656,2003.
[7] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in Proc. IEEE 24th Int. Conf. Data Eng.,2008, pp. 656–665.
[8] R. Hariharan, B. Hore, C. Li, and S. Mehrotra, "Processing spatialkeyword (SK) queries in geographic information retrieval (GIR) systems," in Proc. 19th Int. Conf. Sci. Statist. Database Manage.,2007, pp. 16–23.
[9] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," ACM Trans. Database Syst., vol. 24, no. 2, pp. 256–318, 1999.
[10] Z. Li, K. C. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang, "IRTree: An efficient index for geographic document search," IEEE Trans. Knowl. Data Eng., vol. 99, no. 4, pp. 585–599, Apr. 2010.
[11] N. Mamoulis and D. Papadias, "Multiway spatial joins," ACM Trans. Database Syst., vol. 26, no. 4, pp. 424–475, 2001.
[12] D. Papadias, N. Mamoulis, and B. Delis, "Algorithms for querying by spatial structure," in Proc. Int. Conf. Very Large Data Bases, 1998,pp. 546–557.
[13] D. Papadias, N. Mamoulis, and Y. Theodoridis, "Processing and optimization of multiway spatial joins using r-trees," in Proc. 18thACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst., 1999, pp. 44–55.
[14] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 1998, pp. 275–281.
[15] J. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørva g,
"Efficient processing of top-k spatial keyword queries," in Proc.
12th Int. Conf. Adv. Spatial Temporal Databases, 2011, pp. 205–222.
[16] S. B. Roy and K. Chakrabarti, "Location-aware type ahead
search on spatial databases: Semantics and efficiency," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 361–372.
[17] D. Zhang, B. Ooi, and A. Tung, "Locating mapped resources in
web 2.0," in Proc. IEEE 26th Int. Conf. Data Eng., 2010, pp. 521–532.
[18] D. Zhang, Y. Chee, A. Mondal, A. ung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document,"in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 688–699.